Trajectory Optimization for Mobile Manipulator Motion Planning

Bence Magyar

Heriot-Watt University

School of Engineering and Physical Sciences

A thesis submitted for the degree of

Doctor of Philosophy

August, 2019

HERIOT WATT UNIVERSITY

Research Thesis Submission

Please note this form should be bound into the submitted thesis.

Name:	*Bence Magyar*		
School:	Engineering and Physical Sciences (EPS)		
Version: *(i.e. First, Resubmission, Final)*	Final	Degree Sought:	*PhD*

Declaration

In accordance with the appropriate regulations I hereby submit my thesis and I declare that:

1. The thesis embodies the results of my own work and has been composed by myself
2. Where appropriate, I have made acknowledgement of the work of others
3. The thesis is the correct version for submission and is the same version as any electronic versions submitted*.
4. My thesis for the award referred to, deposited in the Heriot-Watt University Library, should be made available for loan or photocopying and be available via the Institutional Repository, subject to such conditions as the Librarian may require
5. I understand that as a student of the University I am required to abide by the Regulations of the University and to conform to its discipline.
6. I confirm that the thesis has been verified against plagiarism via an approved plagiarism detection application e.g. Turnitin.

ONLY for submissions including published works

Please note you are only required to complete the Inclusion of Published Works Form (page 2) if your thesis contains published works)

7. Where the thesis contains published outputs under Regulation 6 (9.1.2) or Regulation 43 (9) these are accompanied by a critical review which accurately describes my contribution to the research and, for multi-author outputs, a signed declaration indicating the contribution of each author (complete)
8. Inclusion of published outputs under Regulation 6 (9.1.2) or Regulation 43 (9) shall not constitute plagiarism.

* *Please note that it is the responsibility of the candidate to ensure that the correct version of the thesis is submitted.*

Signature of Candidate:	Bence Magyar	Date:	13. April 2020

Submission

Submitted By *(name in capitals)*:	Bence Magyar
Signature of Individual Submitting:	Bence Magyar
Date Submitted:	30. April 2020

For Completion in the Student Service Centre (SSC)

Limited Access	Requested	Yes		No		Approved	Yes		No		
E-thesis Submitted ***(mandatory for final theses)***											
Received in the SSC by *(name in capitals)*:						Date:	13. April 2020				

HERIOT WATT UNIVERSITY

Inclusion of Published Works

Please note you are only required to complete the Inclusion of Published Works Form if your thesis contains published works under Regulation 6 (9.1.2)

Declaration

This thesis contains one or more multi-author published works. In accordance with Regulation 6 (9.1.2) I hereby declare that the contributions of each author to these publications is as follows:

Citation details	B Magyar,N Tsiogkas, B Brito, M Patel, D Lane, S Wang. Guided Stochastic Optimization for Motion Planning. Frontiers in Robotics and AI. 2019;6:105.
Author contributions (using monograms)	BM, BB, and NT contributed conception and design of the work. BM, BB, and MP implemented algorithms and benchmarking. BM and NT performed experiment analysis. BM, BB, and MP wrote the first draft of the manuscript. BM, NT, DL, and SW wrote sections of the manuscript and contributed to several revisions and restructuring. All authors contributed to manuscript revision, read, and approved the submitted version.
Signature:	Bence Magyar
Date:	13. April 2020

Citation details	B. Magyar, N. Tsiogkas, J. Deray, S. Pfeiffer, and D. Lane, "Timed-elastic bands for manipulation motion planning", IEEE Robotics and Automation Letters, vol. 4, no. 4, pp. 3513–3520, Oct. 2019, ISSN : 2377-3766. DOI : 10.1109/LRA.2019.2927956.
Author contributions (using monograms)	BM contributed conception and design of the work and implemented algorithms and benchmarking. SM contributed to benchmarking code. BM and NT performed experiments and analysis. JD contributed Lie-algebra-specific details to implementation and manuscript. BM, NT, and DL wrote the first draft of the manuscript. BM, NT, JD, and DL contributed to revisions and restructuring.
Signature:	Bence Magyar
Date:	13. April 2020

Citation details	J. Deray, B. Magyar, J. Solà, and J. Andrade-Cetto, "Timed-elastic smooth curve optimization for mobile-base motion planning", in 2019 IEEE/RSJ International Conference on Intelligent Robots and Systems (IROS), Nov. 2019, pp. 3143–3149. DOI: 10.1109/IROS40897.2019.8968240.
Author contributions (using monograms)	JD and BM contributed conception and design of the work. JD and JS contributed Lie-algebra-specific details to implementation and manuscript. BM implemented non-Lie components and benchmarking. JD and BM performed experiments and analysis. JD and BM wrote the first draft of the manuscript. JD, BM, JS and JA-C contributed to revisions and restructuring.
Signature:	Bence Magyar
Date:	13. April 2020

Abstract

State-of-the-art robotics research has been progressively focusing on autonomous robots that can operate in unconstrained environments and interact with people. Specifically, manipulation tasks in *Ambient Assisted Living* environments are complex, involving an unknown number of parameters. Recent years show a trend of successfully applied machine learning approaches affecting day-to-day life. Similar tendencies are perceivable in robotics, existing methods being enhanced with learning-based components.

This thesis studies approaches for incorporating task-specific knowledge into the motion planning process that can be shared across a heterogeneous fleet of robots. A step towards data-driven strategies will allow the field to break away from manually-tweaked, heuristics- or state-machine-based solutions and provide good scaling properties, while maintaining operation safety around humans at a very high level.

The presented work proposes a motion planning framework employing *Learning from Demonstration* to encode task-specific motions, facilitating skill-transfer and improving state-of-the-art in motion planning. Resulting algorithms are compared against other methods in a series of everyday tasks.

While different optimisation methods have different benefits, it is possible to build them into systems that both generalise and scale well with the number of tasks and number of robot platforms. This thesis shows that optimisation-based planners are ideal for incorporating prior knowledge into a motion-planning system.

To my wife Évi and my parents, Zoltán and Ildikó.

Acknowledgements

Four years ago, I left my cool robotics job in sunny Barcelona to venture into robotics research in lovely Edinburgh. My PhD studies started in the Ocean Systems Laboratory (OSL) where I was welcomed as a new member of the family.

I would like to thank Prof. David Lane who gave me the opportunity to do a PhD in the OSL in collaboration with the Edinburgh Centre for Robotics, become an independent researcher and fill that blank page as he said during our very first meeting. In addition, I would like to thank Prof. Sethu Vijayakumar and Dr. Vladimir Ivan for patiently discussing many ideas with me during my first year. Many thanks go to Dr. Sen Wang and Dr. Nikolaos Tsiogkas for helping me significantly improve my writing and publication style, as well as, Dr. Valerio De Carolis for his critical thinking on research approaches.

A warm hug to a special group of friends who either became coauthors, participated in bouncing ideas around or simply reminded me to enjoy life too: Mariia, Valerio, Nik, Ingo, Bruno, Corina, Jeremie, Nico, Sam, Hilario and Mayank. Thanks to Ingo, Nik and Jeremie for always being ready with some constructive criticism when I was writing this manuscript.

Very special thanks go to Évi, my wife, for being patient and supporting me through the hardest times of this process and my parents for teaching me that hard work always pays off.

Thank you.

Contents

List of Figures

List of Tables

List of Algorithms

Glossary

SE **S**pecial **E**uclidean groups, whose elements are rigid motions. They combine arbitrary combinations of translations and rotations..

SE(2) SE group covering the 2-dimensional space. Position components are expressed on two orthogonal axes, rotation is defined around third orthogonal axis..

SE(3) SE group covering the 3-dimensional space. Position components are expressed in 3 orthogonal axes, rotation is defined around the same axes..

η A vector of angular velocities $\in \mathbb{R}^3$.

θ Orientation angle used for mobile-bases.

p Pose of a mobile-base in SE(2), top-down representation. It includes a 2D position component and a single angle for orientation.

β A boundary penalty function.

$\mathbb{R}^6$ Task-space, refers to a full 6 DoF specification of a pose in 3D space, also known as SE(3).

X Task-space trajectory. Time series of the *end-effector*'s position defined in SE(3).

$\mathbf{e}$ A collection of error terms.

$\mathbf{\Phi}$ A list of joint positions $\in \mathbb{R}^n$, where n is the number of joints.

G The graph defined by the vertices and edges of the optimisation problem.

$\mathbf{X}_{\textbf{guide}}$ Guide trajectory.

Θ_0 Initial joint-space trajectory of an optimisation-based planner.

T_0 Initial trajectory of an optimisation-based planner. Such methods work by iteratively modifying the initial trajectory according to their cost function(s). This notation leaves the type of the trajectory open, it can be either task- or joint-space..

ϑ A single joint position.

Θ Joint Trajectory. Time series sets of joint states ($\mathbf{\Phi}$-s). It defines the development of joint angles over time.

$d^{\mathbf{L}}$ Mforgot inimum distance.

$\mathbf{p}$ Cartesian 6DoF pose made up of position and orientation elements, $\mathbf{p} \in \mathrm{SE}(3)$.

$\mathbf{X}$ A collection of $\mathbf{p}$ elements.

$\mathbf{r}$ The position component of a pose. Used for both 2D and 3D positions.

$\mathbf{C}^n$ Order of continuity class used to describe a curve or function. Specifically, $f \in \mathrm{C}^n$ means that f's n-th derivatives are continuous..

$\mathbf{w}$ A vector of weights.

List of Abbreviations

AABB *Axis-Aligned Bounding Box*

AAL *Ambient Assisted Living*

AUV *Unmanned Aerial Vehicle*

CHOMP *Covariant Hamiltonian Optimization for Motion Planning*

CMA-ES *Covariance Matrix Adaptation Evolution Strategy*

CLAMP *Combined Learning from demonstration And Motion Planning*

DMP *Dynamical Movement Primitive*

DoF *Degrees of Freedom*

DTW *Dynamic Time Warping*

EDG *Euclidean Distance Grid*

FCL *Flexible Collision Library*

FK *Forward Kinematics*

G2O *General Graph Optimization*

GMM *Gaussian Mixture Model*

GMR *Gaussian Mixture Regression*

GPMP2 *Gaussian Process Motion Planner 2*

GPMP *Gaussian Process Motion Planner*

GP *Gaussian Process*

GSTOMP *Guided Stochastic Optimization for Motion Planning*

IK *Inverse Kinematics*

KDL *Kinematics and Dynamics Library*

KL divergence *Kullback-Leibler divergence*

LBS *Linear Block Solver*

LfD *Learning from Demonstration*

MAP *Maximum A Posteriori*

MPC *Model Predictive Control*

NURBS *Non-Uniform Rational B-splines*

OG *Occupancy Grid*

OMPL *Open Motion Planning Library*

PbD *Programming by Demonstration*

PF *Potential Field*

PI^2 *Policy Improvement with Path Integrals*

PRM* *Probabilistic Road Maps**

PRM *Probabilistic Road Maps*

ProMPs *Probabilistic Motion Primitives*

RL *Reinforcement Learning*

ROS *Robot Operating System*

RRT* *Rapidly-exploring Random Tree**

RRTConnect *Rapidly-exploring Random Tree Connect*

RRT *Rapidly-exploring Random Tree*

RAMP *Real-time Adaptive Motion Planning*

SBPL *Search-Based Planning Library*

SDF *Signed Distance Field*

SQP *Sequential Quadratic Programming*

STOMP *Stochastic Trajectory Optimization for Motion Planning*

TEB2MP *Timed Elastic Bands for Manipulation Motion Planning*

TEB *Timed Elastic Bands*

TESC *Timed-Elastic Smooth Curve*

TRAC-IK *TRACLabs Inverse Kinematics Solver*

TrajOpt *Trajectory Optimization for Motion Planning*

UAV *Autonomous Underwater Vehicle*

PCA *Principal Component Analysis*

Thesis' Publications

During the course of the PhD studies whose outcome is this work the following publications have been made:

S. Chitta, E. Marder-Eppstein, W. Meeussen, V. Pradeep, A. R. Tsouroukdissian, J. Bohren, D. Coleman, B. Magyar, G. Raiola, M. Lüdtke, *et al.*, "Ros_control: A generic and simple control framework for ROS", *The Journal of Open Source Software*, vol. 2, p. 456, 2017.

B. Magyar, N. Tsiogkas, B. Brito, M. Patel, D. Lane, and S. Wang, "Guided stochastic optimization for motion planning", *Frontiers in Robotics and AI*, vol. 6, p. 105, 2019.

B. Magyar, N. Tsiogkas, J. Deray, S. Pfeiffer, and D. Lane, "Timed-elastic bands for manipulation motion planning", *Intelligent Robots and Systems (IROS), 2019 IEEE/RSJ International Conference on*, 2019.

B. Magyar, N. Tsiogkas, J. Deray, S. Pfeiffer, and D. Lane, "Timed-elastic bands for manipulation motion planning", *IEEE Robotics and Automation Letters*, vol. 4, no. 4, pp. 3513–3520, Oct. 2019, ISSN: 2377-3766. DOI: 10.1109/LRA.2019.2927956.

J. Deray, B. Magyar, J. Solà, and J. Andrade-Cetto, "Timed-elastic smooth curve optimization for mobile-base motion planning", in *2019 IEEE/RSJ International Conference on Intelligent Robots and Systems (IROS)*, Nov. 2019, pp. 3143–3149. DOI: 10.1109/IROS40897.2019.8968240.

Chapter 1

Introduction

MANIPULATION tasks remain a challenging problem for mobile robots in human environments. Currently, the largest robot workforce can be found in automation, the car manufacturing industry spearheading the deployment of automated solutions, with a rising tide of autonomous warehousing robots following. Standard application setups in these fields, however, are strictly industrial, with a fixed set of tasks (often a single one) and very limited flexibility in task parameters.

Ambient Assisted Living (AAL) became a focus in robotics early on, as social studies revealed how maintaining independence and a good quality of life in aging societies is going to be a significant, multi-factor issue. Robots working in home environments are often expected to react to spoken commands, locate and collect certain objects from open shelves or closed storage compartments, detect and address emergencies by calling for assistance. For true applied AAL to become a reality, systems that can handle an extremely wide range of environments as well as a variety of tasks in an online fashion is of paramount importance. The acquisition of such skills, however, is - more often than not - done by very specific, carefully crafted mathematical models and behaviours. For instance, Rühr *et al.* in [1] demonstrated opening a wide range of cabinets, drawers and doors in a kitchen environment with a mobile manipulator. The proposed approach is a control framework composing handle pose detection through vision and a continuous estimation of the articulation model of the current furniture piece. Visual servoing approaches have been popular for manipulating everyday furniture. Similarly to other control-based approaches to manipulation however, they operate with a model of the manipulated objects, the least is being visual cues. The recent work of Paolillo *et al.* [2] tackles opening drawers, a printer and a cabinet. Visual servoing typically provides

Figure 1.1. *Ambient Assisted Living* (AAL) is a collection of concepts, products and services which combine new technologies and the social environment in order to improve quality of life in all periods of life. Image source: [3]

robust solutions when executed on real robots thanks to the direct feedback loop between perception and the manipulation engine. A common limitation however is that it requires some model of the objects (sometimes complete 3D) and the solutions are tied to the control framework, the robot's perception setup and the specific tasks. To improve this, robots must have the ability to learn and execute new behaviours to achieve desired tasks as they assume collaborative roles alongside humans in their unstructured, non-robot-optimised environments.

The work presented in this thesis provides motion planning solutions for robots able of fulfilling AAL and small to medium-scale warehousing tasks. The *RoboCup@Home* league is added to the popular international robotics initiative, *Robocup*, in 2006. Competitions drive development of service and assistive robot technology for future personal domestic applications. It quickly became the largest international annual competition for autonomous service robots. The ultimate scenario is the real world itself. Each year, the bar is raised by a little, gradually building up required technologies to handle tasks in a basic home environment. Specifically, such technologies focus on manipulation, social behaviours, navigation, mapping and object detection. In the first years, the scenario consisted of a living room and a kitchen but soon it was followed by other areas of daily life, as well as tasks of increasing complexity. Robots are expected to understand tasks from speech, navigate the apartment, find objects of interest, manipulate furniture, pick, transport and place objects and return sensible answers to queries from their human masters.

Numerous manufacturers and research labs designed mobile manipulator robots to

Figure 1.2. Willow Garage's PR2 [4], Fraunhofer IPA's Care-O-bot ®[5] , Fetch [6] from Fetch Robotics, Cosero [7] from the University of Bonn, PAL Robotics' *TIAGo* [8] and Toyota's HSR [9]. Image source: [10]

meet the challenges presented in this domain. A non-comprehensive set of these robots is shown in Figure 1.2.

The RoboCup competitions showed that there is not a single best platform that triumphs at all challenges. Likewise, in a future with robot helpers in homes, it is hard to imagine that a single robot platform would fit everyone's needs in the foreseeable future. Rather, similarly to how the automotive industry works, different providers will compete offering different tiers of robot capabilities. However, it is desirable that various platforms share a similar technology and software stack. A full-blown robotics product is an order of magnitude more complex than any automotive product both at the hardware and software level. Naturally, the last 12 years of mobile robotics research and industry moved towards shared, open-source middleware libraries. The popularity and usage statistics of *Robot Operating System* (ROS) [11] confirms this trend [1].

Hence, this work focuses on the manipulation motion planning aspect of the AAL challenges as well as keeping solutions applicable to small to medium-scale logistics. Planning task-specific motion, while providing smooth robot arm movement has been solved before, however, a key to an effective system for such challenges is one that can scale to multiple tasks.

There is a large variety of trajectory optimisation methods that offer tools to minimise some cost functions while maintaining a set of constraints. It solves an optimal control problem with an open-loop algorithm. Closed-loop, control-based approaches may propose operating kitchen furniture by making some assumptions on the model and estimating free parameters on-the-fly during the interaction. For instance, the approach presented in [13] handles multiple models simultaneously, assuming a single manipulable axis and refines the initial estimation via force-torque sensors. While the approach

[1] Over 16.000 downloaded packages only from the official repositories [12]

may prove robust for the already modeled types, extending this list requires careful modeling. Leveraging prior knowledge of experts instead of manually designed task-specific solutions is desirable. Opening doors and using different knobs were shown by Dang *et al.* in [14] using human demonstrations. First, objects and humans are instrumented with tracking markers in key locations, then a clustering algorithm is used to partition motions into segments of rotational and translational components. The extracted manipulation primitives are then put back together in their original order forming a manipulation chain. In order to use such manipulation chains with novel objects, the same set of markers need to be attached in an equivalent fashion which makes the practical application of this approach labour-intensive.

The field of *Learning from Demonstration* (LfD) offers different, learning-based methods for solving this challenge while there exist motion planner algorithms that can incorporate prior knowledge into the planning process. Given the breadth of available robot platforms for these domains, an ideal system would also not depend on robot-specific parameters, allowing for a wide adoption to a variety of robot platforms.

As an application scenario for the scope of this work, basic furniture manipulation and picking tasks will be used. This decision is motivated by the choice of target domains of AAL and small to medium-scale logistics.

1.1 Research objectives

This work aims at expanding the field of robot motion planning by leveraging *Learning from Demonstration* (LfD) and also ensuring a level of safety (limits, smoothness of movement, etc.). Specifically, this work tackles the following questions:

- How can LfD be used for robot motion planning?
- How can trajectory optimisation be used to generalise across multiple robots with different kinematic structures?
- Are solutions applicable to real-world, online operation?
- Which type of properties do these methods have?

These questions are relevant for the advancement of the motion planning field in the AAL context. The goal is to provide solutions that can be applied online and achieve a level of scaling both in terms of number of tasks and robot platforms. Such a system is ideal for mobile manipulators intended to work in home environments and small to medium-scale warehousing operations. Additionally, a desired property of such a system is using a stable set of parameters to avoid situations where for every use-case, internal parameters need to be tuned. This aspect will be key when designing planning approaches in this thesis.

Throughout this thesis a *task* refers to a general concept of a manipulation problem, e.g. opening a drawer. Such tasks have variants depending on how many and what kind of open parameters they have, a specific parametrization of a task is referred to as a *task instance* and sometimes for brevity may also be referred to ask a *task*.

1.2 Methodology

Initially, several frameworks and methods for LfD are studied. For representing learned movements, a Cartesian-space description is found necessary as this representation can directly encode the movement of the *end-effector* in 6 *Degrees of Freedom* (DoF). The common alternative representation is to use joint-space, encoding motion in robot state-space which binds such information to the kinematic structure of a specific robot platform. Following up, a way of combining *Stochastic Trajectory Optimization for Motion Planning* (STOMP) and LfD is attempted using demonstrations of pick and place and

drawer- and cupboard opening on real robots. An analysis of this system regarding scaling for these tasks using multiple simulated robots and varying degrees of freedom in the manipulator system is created. Once confirmed that the system is able to bring task-specific knowledge into optimisation-based motion planning algorithms, general planning quality is compared against state-of-the-art planners. The setup is interfaced and validated with actual robots in realistic environments at the Robotic Independent Living Laboratory at Heriot-Watt University, Edinburgh and at the Warehouse and Retail Logistics Laboratory at Fraunhofer IPA, Stuttgart.

The next phase of this work is to introduce additional quality objectives to the optimisation process and validate changing the planner module in support of this. Handling safety around people is an important aspect of a mobile manipulator. Desired properties from such a robot are smooth, predictable movements, at a moderate speed with reasonable reaction time. Smooth trajectories are n-derivable and have bounded velocity, acceleration and jerk. The STOMP planner did not yield reliable smoothness properties, requiring very careful tweaking depending on the task, the specific task-instance and the robot platform. Previous work [15] has shown that *Least Squares-based* optimisation serves as a viable approach to motion planning. Furthermore, the parametrization of the optimisation step with this approach is determined during the process, the gradient guiding it to convergence much more reliably, resulting in more consistent results than STOMP. This phase proved that a planner built with *Least Squares-based* optimisation can handle a series of trajectory properties providing reasonable safety features. A mobile-base motion planner is created and compared against a state-of-the-art method showing the required improvement and consistency in quality.

Finally, an arm motion planner is created by extending the mobile-base planner presented in the previous phase. Smoothness properties defined earlier are extended to handle 6 DoF task-space smoothness. The proposed system is shown to be able to handle error functions defined in different spaces. Additionally, as this planning process follows a similar setup as the STOMP planner from the first phase, it is possible to incorporate task-specific knowledge the same way. The proposed approach is compared against state-of-the-art planners, including STOMP, showing a good, consistent performance without any need for adjusting parameters between tasks and task instances.

1.3 Contributions

In the scope of this work, the following contributions are made:

1. Extend the STOMP motion planner with flexibly retargeted trajectories acquired via LfD [16] and achieve scaling over number of tasks and different task instances.

2. Achieve skill-transfer between robots [16].

3. Formulate a graph-based motion planning problem for mobile-bases that is able to enforce smoothness and a set of other properties [17].

4. Introduce a novel, graph optimisation-based arm motion planner that provides consistently good performance and consistently smooth output trajectories [18], [19].

The results collected through this work show a solution for a task-informed motion planning system that is both scalable in terms of tasks and robot platforms while maintaining a level of safety via smooth, predictable output paths.

1.4 Structure

The rest of the work is organised as follows. Chapter 2 presents relevant background, a detailed literature review of *Learning from Demonstration* (LfD) and motion planning problems and a description of the robot platforms used in this work.

In Chapter 3, a motion planner merging LfD and stochastic optimisation is introduced with a system architecture that supports multiple skills and skill transfer between robot platforms. A rich set of comparisons are made against state-of-the-art planners and discussed in detail. Further insight is provided into this specific optimisation and LfD method.

Chapter 4 presents a formulation for generating *smooth trajectories* in a graph optimisation process for mobile-base planning. The approach is validated on a mobile-base path-planning problem and is compared against a state-of-the-art, industry-proven planner available in ROS.

Chapter 5 builds upon the work presented in Chapter 4, essentially extending the planning scheme to arm planning. The difference in domain - e.g. mobile-base path

planning and arm planning - make for a significant increase in complexity. The formulation for smoothness is extended to the more complex 3D scenario, the defined limit constraints are extended to include joint- and Cartesian-space limits as well as an entirely different distance metric for collision avoidance, to account for solving the much harder, higher dimension problem.

Finally, Chapter 6 summarises the contributions of this work. It discusses limitations and provides future research directions.

Chapter 2

Background

This chapter presents relevant background knowledge: trajectory representations, *Learning from Demonstration* (LfD) and motion planning from the literature, as well as the robot platforms used in the experiments presented in this thesis. Section 2.2 provides conceptual insight into different trajectory representation methods and their relationship to forward and inverse kinematics. Section 2.3 presents existing LfD approaches while Section 2.4 details approaches addressing the motion planning problem.

2.1 The Motion Planning Problem

Motion planning is one of the most important aspects of any robotic system as it enables its interaction with the real world. Operating in the real world requires fast and efficient methods that find plans for the robot to achieve its goals.

In general the motion planning problem requires the robot to reach a goal from a starting configuration. A set of additional requirements are often necessary such as planning without colliding with obstacles or the robot itself, respecting robot joint limits, *end-effector* limits, smoothness and other constraints. Joint limits can, for instance, disallow movements that would over-twist or break joints of a robot arm while smoothness achieves a level of predictability of robot movements and allows people to feel safe around the robot.

There exist many options for specifying starting and goal configurations and the motion planning process may use different representations to operate on internally.

2.2 Trajectory representations and kinematics

A joint space trajectory representation Θ is defined as a time series of joint state vectors $\mathbf{\Phi}$ such that $\Theta = [\mathbf{\Phi}_1, \ldots, \mathbf{\Phi}_n]$. It defines robot motion through the development of the position of each joint. The units of individual fields in a joint state vector $\mathbf{\Phi}$ are not always homogeneous. Depending on the type of joint the position field corresponds to, it will be one of the following units:

- angle (in radians) for revolute joints
- distance (in meters) for linear joints

Screw joints are not commonly handled in such frameworks and spherical joints are decomposed into three virtual revolute joints. The benefit of a proper joint-space trajectory is that the format does not require further processing, it is directly relatable to and possibly executable on a robot. To reason about task-space constraints is straightforward, one only has to use prior knowledge of the kinematic structure and solve a *Forward Kinematics* (FK) problem to compute the position of a given point on the robot given a joint state vector $\mathbf{\Phi}$. The drawback of this representation is that it is fully tied to the kinematic structure of the robot. A joint trajectory computed for one robot will only work on robots of the exact same kinematic structure.

Task-space trajectory representations $\mathbf{X}$ define the pose $\mathbf{p} \in \mathrm{SE}(3)$ of a point of interest and its development over time such that $\mathbf{X} = [\mathbf{p}_1, \ldots, \mathbf{p}_n]$. Such a point is most often an important place on the *end-effector*, for instance, between the parallel jaws of a gripper. The benefit of a task-space trajectory is that it can describe a manipulation task entirely in its own domain, without being tied to robot-specific representations. Task-space trajectories require a conversion to joint space for execution called the *Inverse Kinematics* (IK) problem.

Both trajectory representations above may choose to include velocity and acceleration components. This is considered an implementation detail and is computable from the available position and timing information.

This work is not focusing on the kinematics problem but it is important to note that FK and IK are used extensively to convert from joint space to task space and task space to joint space, respectively. There is a rich literature on FK and IK and widely available implementations of different IK solvers. One of the most popular IK solvers is provided by the *Kinematics and Dynamics Library* (KDL) [20] and uses an joint-limit-

constrained pseudoinverse Jacobian approach. To account for false-negatives exhibited by the KDL solver on complex robot platforms, [21] proposed *TRACLabs Inverse Kinematics Solver* (TRAC-IK). It employs a multi-threaded approach, where both KDL and a *Sequential Quadratic Programming* (SQP)-based inverse kinematics solver are run. This solution takes advantage of the speed KDL provides when it is able to solve the problem and the reliability, albeit slight speed reduction which comes from the more complex, optimisation-based SQP solver. The approach presented by Vahrenkamp *et al.* in [22] acquires solutions via a reachability map, hence the name of their method, *IK-MAP*. An alternative approach known for its efficiency is the *IK-FAST* algorithm [23] which operates on kinematic chains with up to 6 *Degrees of Freedom* (DoF). Manipulator systems with more DoF can be worked around by operating only on a subset of active joints, while virtually freezing the rest.

2.3 Learning from Demonstration Literature

Intuitively, the paradigm for enabling robots to acquire knowledge for completing new tasks is not unique. Depending on the sub-field, one may find references to this paradigm under the names of (Robot) *Learning from Demonstration* (LfD), (Robot) *Programming by Demonstration* (PbD), Apprenticeship Learning and Imitation Learning. This thesis will refer to this principle as *Learning from Demonstration* (LfD).

Before the start of LfD in the 1980s, robots had to be carefully hand programmed for every task they performed. To eliminate this practice, LfD methods aim to solve the difficult step of letting users train their robots. A naturally occurring requirement for this is keeping the training stage user-friendly, allowing robots to be utilised to a larger extent in daily interactions without requiring special operator training. Furthermore, these demonstrations provided by users serve as expert knowledge, further enhancing the efficiency of the learning process compared to trial-and-error learning, especially in high dimensional systems.

As opposed to carefully modeling specific problems, LfD approaches offer leveraging observations of a human's performance and deriving an appropriate robot controller from it. The main goal is to provide a more scalable approach for acquiring and adapting robot capabilities to novel situations, in a form accessible without robot programming knowledge as well.

Billard *et al.* surveyed robot learning methods in [24], featuring many early ap-

proaches among which the work of Calinon *et al.* [25] has proven to become the most impactful. The LfD framework proposed in their paper uses *Principal Component Analysis* (PCA) to project the motion data onto latent space and train a *Gaussian Mixture Regression* (GMR) model to generalize and adapt the learned trajectories. The approach was validated on a 9 DoF system in tasks involving moving a chess piece, bringing a two-handed bucket to a specific position, picking a sugar cube and moving it to the mouth of the robot. Many modern approaches based on learning and adapting trajectories took inspiration from this work.

Reachability Maps and Inverse Reachability Maps were explored in [26] and [27] respectively. Both works explore the placement of a mobile manipulator to execute stored Cartesian trajectories for the task of drawer opening and pick and place respectively. Despite targeting at similar tasks as this work, the main focus on these approaches is on a single task at a time. They try to solve each task by placing the mobile base in an appropriate position that would allow the execution of a task-specific trajectory. Opposite to what many other methods in the literature do, these two approaches do not focus on generating the trajectory required to execute the task.

Kyrarini *et al.* [28] proposed to use *Gaussian Mixture Model*s (GMMs) to learn a motion from several human demonstrations. They performed kinesthetic teaching to record the demonstrations. Despite the presented results that method was demonstrated in a setup tailored to solving a specific task. Unfortunately, there was no variation to the task being solved to show that it can generalise well. In addition, the presented approach is a collection of pre-existing tools instrumented to solve one specific task, adding small value to the state-of-the-art.

*Dynamical Movement Primitive*s (DMPs) [29] present a time-independent, scalable trajectory representation that allows start and end states to be changed while maintaining the dynamic characteristics of the motion used as demonstration. DMPs are able to represent motion in either joint space or Cartesian-space although rotations in the latter case require special attention as discussed in [30], [31].

Niekum *et al.* presented a series of algorithms [32] building on Bayesian non-parametric statistics and control theory. Specifically targeting robust generalisation capabilities to automatically detect and leverage repeated structure in the training data. One such type of abstract information possible to extract is whether a skill is appropriate for a task, certain task invariants, features of importance and high-level task structure. This culminates in the generation of a finite state machine consisting of building blocks

of grounded skills implemented with DMPs. They evaluated these algorithms using a *PR2*[1] mobile manipulator.

A unified representation for motion is proposed in [33]: the output of marker-based motion-capture is converted into a unified representation called the *Master Motor Map* which converts the motion in form of a DMP to any humanoid robot known to the system.

Policy Improvement with Path Integrals (PI^2) [34] is a probabilistic learning algorithm with a single parameter as exploration noise. It scales well to high dimensions and ideal for optimising joint space DMPs. The work presented in [35] used PI^2 for learning policies over sequences of DMPs for grasping under uncertainty to solve pick and place tasks. A thorough review of the family of algorithms including *Covariance Matrix Adaptation Evolution Strategy* (CMA-ES), PI, PI^2, PI^{BB} and *Stochastic Trajectory Optimization for Motion Planning* (STOMP) is presented in [36].

Paraschos *et al.* studied different Movement Primitive frameworks in [37] with the aim of establishing a unified framework that implements all common features in a principled way. The proposed new representation, coined *Probabilistic Motion Primitives* (ProMPs), includes a data-driven representation of movements and supports generalisation to new situations, allows temporal modulation, sequencing of primitives and controllers for executing the primitive on physical systems. They show promising results on both simulated and real-robot scenarios.

Paxton *et al.* in [38] proposes an iterative learning framework to resolve some of the limitation of DMPs, namely the limited ability to generalise to new environments, to solve obstacle avoidance and to cope with inherently noisy human demonstrations. A GMM module mimics paths in new environments, DMPs generalise to different start and end positions. An *Inverse Optimal Control* module is used to compute the reward function which is then used to construct a control policy achieving sweeping tasks in their evaluations. The concept corresponds to different types of observational learning from the field of developmental psychology.

Rey *et al.* in [39] adapted the PI^2 algorithm to a time-invariant Dynamical System using GMR. A set of demonstrations are required to train the GMR model. The time-invariant Dynamical System is used as a parametrised policy which PI^2 operates on in a *Reinforcement Learning* (RL) fashion. The search algorithm handles the sampling of policy parameters as well as ensures stability of the resulting motion. A state-dependent

[1] https://robots.ros.org/PR2/ (Last accessed: August, 2019.)

Table 2.1. Summary of key surveyed LfD methods and their features.

Name	Type	Space	Online	Motion Generalisation	Ref.
Reachability Map (RM)	Lookup method	Task	Yes	No	[26]
Inverse RM	Inverse planning	Task	No	No	[27]
GMR	Regression	Task	Yes	Yes (no domain info)	[25]
DMP	Dynamic model	Task/Joint	Yes	Yes (many flavours)	[29]
PI^2	RL	Joint	No	Yes (embedded model)	[39]
ProMPs	Statistical model	Task/Joint	Yes	Yes (many flavours)	[37]

stiffness model is also produced in conjunction with the policy which together form the core of an impedance control architecture handling execution of tasks. The learning architecture is validated on a *KUKA LWR robot arm* accomplishing a digging task.

A method for robust sequencing of Motion Primitives was presented by Lioutikov *et al.* in [40] by combining probabilistic context-free grammars with a library of primitives. The rule-based nature of formal grammars is ideal for search methods since it allows generating complex motion policies by encoding both hierarchically and recursively structured tasks. A Markov Chain Monte Carlo optimisation is used to acquire the grammar from observations. The result is a distribution over the probabilities of certain operators connecting the search space. Continuity can be achieved by applying restrictions to the grammar. The method was successfully demonstrated on a 7 DoF lightweight robotic arm playing tic-tac-toe and completing a box assembly task.

This section discussed state-of-the-art LfD methods relevant for use in a motion planning framework. The key features of the studied approaches are summarised in Table 2.1. The DMP and ProMPs approaches stand out as they are both able to provide a minimum-dependency, robot-agnostic representation module for incorporating and reproducing learned movements. In conclusion, a task-space DMP implementation was chosen to represent task-specific motion due to its flexibility of only requiring a single demonstration as well as being robot-agnostic. As it will be explained in Chapter 3, the proposed motion planning framework only relies on an abstract interface to the LfD component and could be replaced with a ProMPs-based module if required in the future.

2.4 Motion Planning Literature

The recent years have seen a raise in successfully applied machine learning approaches in many fields affecting day-to-day life. A similar trend is perceivable in robot motion planning: different type of function approximators (regressors, neural networks,

etc.) started replacing meticulously hand-crafted models and heuristics as well as optimisation-based methods (numerical optimisation, RL, etc.) are being used to come up with motion plans by iteratively improving on an initial solution. While the time of fully learned, end-to-end motion planners has not come yet, existing algorithms can leverage different learning-based methods, depending on their requirements and constraints.

A common phenomenon in motion planning is local minima. Optimisation- and search-based planners are both subject to this, when the planning process cannot see far enough beyond the current suboptimal solution or the evaluation of objective functions balance each other out.

The problem of motion planning has been actively researched in the past decade. *Rapidly-exploring Random Tree*s (RRTs) [41] are state-of-the-art planning algorithms highly efficient and widely used for low-dimensional problems. RRTs excel at exploring large search spaces thus are often employed for path-planning on mobile robots, autonomous cars or robotic arms. *Rapidly-exploring Random Tree Connect* (RRTConnect) [42] and *Probabilistic Road Maps* (PRM) [43] are sampling-based methods that are able to generate trajectories in complex and high dimensional spaces. Despite their ability to generate feasible trajectories fast, the quality of the produced solution is often poor. Early randomisation-based approaches such as PRM struggled with general nonholonomic and kinodynamic planning problems. These methods use randomness in the search procedure which often leads to redundant or jerky motions reducing solution quality. To alleviate this, a post-processing step is often applied which, depending on the method, may cancel out the gain of efficiency of the search algorithm. As the solution provided by the aforementioned sampling-based motion planners is usually non-optimal, new asymptotically optimal planners have been introduced in the work of [44]. This work provides new methods, specifically the *Rapidly-exploring Random Tree** (RRT*) and *Probabilistic Road Maps** (PRM*) algorithms, that improve on finding the optimal path given a cost metric.

Even though sampling-based methods are able to find optimal paths, the provided solutions can potentially violate constraints imposed by the robot hardware. For example, solutions may not be feasible since they may not consider all kinematic and dynamic constraints. Recently some extension to RRT* did incorporate kinematics considerations for simple models such as a mobile-base [45]. Trajectory optimisation techniques have been introduced to impose constraints directly in the planning pro-

cess. These methods take an initial trajectory as an input and optimise it based on a set of cost functions. This initial trajectory can be generated by the aforementioned sampling-based methods, interpolation or other means.

Initially designed for obstacle avoidance, the *Potential Field* (PF) method proposed by Khatib *et al.* in [46] proved to be very valuable for trajectory planning. Two artificial potential fields are superimposed, one repulses from obstacles while the second attracts toward the desired goal. While being simple and efficient, its main drawbacks are the presence of local-minima in which the robot gets stuck and typically fails to find a path between close obstacles.

Covariant Hamiltonian Optimization for Motion Planning (CHOMP) [47], [48] and its variants [49], [50] employ an optimisation-based approach which works by iteratively improving an initial trajectory. Covariant Gradient Descent is used to optimise along a cost function responsible for smoothness and obstacle avoidance. CHOMP was included in almost all experiments in this thesis, however it was removed from the results in many cases due to its quality of results. This algorithm proved very sensitive to parameters and non-differentiable cost functions, making it very hard to build upon for a general-purpose system. Building on CHOMP the work presented in [51] uses CMA-ES and a distribution of demonstrations to learn cost functions for CHOMP. This work presents appealing results in task instances of disentangling a rope while avoiding collision with objects. However, it needs a rich set of joint space demonstrations for every task and it is tied to the joint-space representation of motion.

STOMP, presented in [52], performs optimisation on non-differentiable constraints by drawing samples stochastically from a set of noisy trajectories. Unfortunately, these methods can potentially be computationally expensive as they require a finely-discretised trajectory to perform obstacle collision checks and guarantee a smooth solution. At the same time they may fail to converge on even moderately hard problems. In addition, the performance of STOMP is strongly dependent on the parameters used for noise generation. In many cases a set of parameters will produce good results, while the same set will perform poorly for a different problem.

To avoid the potential computational complexity of STOMP, the *Trajectory Optimization for Motion Planning* (TrajOpt) method was introduced as a solution in [53]. It defines the optimisation problem as a *Sequential Quadratic Programming* (SQP) problem with continuous-time collision checking. The reduced computational cost is achieved by using a sparse solution, where a trajectory is represented by a small number

of states and using continuous-time collision checking. In addition, the SQP formulation allows hard constraints to be imposed such that the produced plan is guaranteed to respect them. TrajOpt was also proven versatile in a series of different tasks presented in [54] and was shown to consistently outperform CHOMP.

Experience-based planning with *sparse roadmap spanners* [55] relies on a large *experience database* of pre-recorded whole-body trajectories of a given robot. It aims to use these as a start of the optimisation and *repair* them according to a new scene. These experiences are used to encode solutions to some computationally expensive checks, such as self-collision, joint limits and stability constraints. Roadmap spanners focus on high DoF, highly constrained systems such as bipedal humanoid robots and operates on experiences recorded in joint space.

The work of Phillips *et al.* in [56] aims to accelerate manipulation tasks by means of a combined planning approach of the mobile-base and a manipulator. Concretely, an Experience Graph (EG) is built using demonstrations acquired via kinesthetic teaching, then a weighted A^* search is used to find solutions over the EG. Obstacle avoidance for novel objects is achieved by the weighting mechanism in conjunction with multiple demonstrations. The method was validated on simulated and real scenarios using a *PR2* robot on tasks involving articulation constraints, e.g., drawer and cupboard opening, where the motion of the object itself involves a single DoF. The results showed a significant improvement compared over a naive weighted A^*.

The work of [57] proposes a joint space trajectory representation using *Reproducing Kernel Hilbert Spaces*. They elaborate on the optimisation process and update rules with respect to smoothness and obstacle avoidance constraints and show to perform better than CHOMP in a simulated scenario. There is a strong motivation for using such representations as they make reasoning about smoothness, described by acceleration, jerk, snap etc., trivial. Unfortunately, the many open parameters make this method hard to tune for any practical application. The authors propose using different kernels which all come with their respective parameters on top of the parameters of the proposed optimisation method. Unfortunately, there was no suggested method to tune the parameters, making it hard to apply for a different problem or even replicate the results. Moreover, adding additional constraints in a gradient-dependent system requires adapting the optimisation process itself.

Cohen *et al.* proposed *Search-Based Planning Library* (SBPL) in [58], a search-based method where a graph is built by using atomic motions referred to as motion prim-

itives. The primitives in SBPL serve a guiding role, exploring the state-space reachable by the robot kinematics. However, they are only static building blocks, being limited to a certain set of moves. Compared to DMPs they lack flexibility and only encode single steps rather than entire trajectories. Generally, lattice planners have been widely used for problems where decision-making and the effects of decisions are also considered in the planning process. Browne *et al.* provide a good summary of this approach in [59].

DeBaTo [60] employs ProMPs in a Relative Entropy Policy Search framework for obstacle avoidance and trajectory optimisation. A method to compare generated trajectories to a set of demonstration trajectories was used during the optimisation process. This approach featured the distribution-based *Kullback-Leibler divergence* (KL divergence) as distance measure in the optimisation against the set of demonstrations. In principle, a distribution-based approach is very useful, however, it also operates with the strong assumption of having multiple, statistically diverse - yet meaningful - demonstrations of task instances available.

Stark *et al.* [61] extended the above idea by allowing learning of new skills from previously solved task instances. Tasks are defined by a combination of joint space ProMPs and a description of their effects. When learning a new skill, it is first initialised with parameters inferred from related movements, then iteratively adapted using Relative Entropy Policy Search. Demonstrations on a 3 DoF manipulator showed success on a pushing task. This approach may yield very interesting results when combined with a *roadmap spanner* framework where joint space experiences form the core of the task knowledge.

The method proposed in [62] integrates joint space ProMPs with the CHOMP algorithm. Rana *et al.* adopt a probabilistic approach to motion representation in joint space, however, the adopted probabilistic approach requires several, statistically diverse demonstrations of a given task.

The *Timed Elastic Bands* (TEB) planner [15], [63] quickly became one of the most popular motion planners for mobile-base navigation in the *Robot Operating System* (ROS) [11] community. Based on the ideas originating from Quinlan *et al.* [64], this planner allows efficient and rapid estimation of a discretised trajectory in the plan. It incorporates constraints such as kinematic feasibility considering the properties of the current robot platform, as well as, velocity limits and obstacle avoidance using a multi-objective optimisation formulation. Although this type of optimisation has many parameters, they are typically used to balance error terms coming from different sources,

a process that does not need adjustments depending on the scenario the robot is solving. The problem is defined as a nonlinear optimisation program which may be solved by nonlinear least-squares optimisation techniques.

The idea of formulating an arm motion planning task as a graph-optimisation problem was first discussed by Dong *et al.* in [65]. The adopted representation considers the trajectory a continuous valued function that maps time to robot states. Trajectory optimisation is performed using probabilistic inference. A *Gaussian Process* (GP) is used to provide a prior function that encourages smoothness. Collision free trajectories are encouraged by a likelihood function. The posterior distribution of the GP prior with the likelihood function is used to calculate the *Maximum A Posteriori* (MAP) estimate of the trajectory. This method, coined *Gaussian Process Motion Planner 2* (GPMP2) provided comparable success rates with the state-of-the-art while requiring much less computational effort. Despite the presented benefits the practical deployment of GPMP2 is limited in various ways. Joint limits are respected only by clamping their maximum values in the initial trajectory. This can lead to situations where the found trajectory may not be valid after the clamping operation and there is nothing explicitly encouraging the optimiser to respect joint limits. Moreover, trajectory smoothness is only encouraged by using a prior having no acceleration. While this may be enough in basic cases, it does not guarantee that the final generated trajectory will be smooth. The presented method does not consider the generated trajectory length in either euclidean or joint space , nor it allows for the introduction of waypoints to reach in various stages of the generated trajectory.

Bhardwaj *et al.* [66] incorporated past experience in order to learn automatic tuning of *Gaussian Process Motion Planner* (GPMP) algorithm parameters. The main motivation behind this work was to alleviate parameter tuning in optimisation-based planners and provide a data-driven, end-to-end parameter tuning algorithm. Specifically, the work studied how to extend GPMP2 to be differentiable. As these parameters play a crucial role in the quality of results, a data-based tuning approach may yield results that allow methods suffering from such limitations triumph.

The TrajOpt[53] method was introduced to avoid the potential computational complexity of CHOMP(2.4) and STOMP. To leverage the toolkit of classic numerical optimisation, it proposes to formulate the task as a SQP problem. A sparse representation ensures that computational cost is reduced while a continuous-time collision checking ensures safety. In addition, the SQP formulation allows hard constraints to be imposed

such that the resulting plan is guaranteed to respect them. TrajOpt was also proven versatile and consistently superior to CHOMP in a series of different tasks presented by Schulman *et al.* in [54].

The idea for employing alternative trajectory representations was the interest of many researchers in motion planning. There is a strong motivation for representations that ease reasoning about smoothness, for instance, the joint-space trajectory representation using *Reproducing Kernel Hilbert Spaces* proposed by Marinho *et al.* [57]. Unfortunately, the many open parameters make this method hard to tune for any practical application. Authors proposed different types of kernels which all come with their respective parameters additionally to the parameters of the proposed optimisation method.

Boutselis *et al.* [67] developed a novel optimal control framework for uncertain mechanical systems. The proposed algorithm handles uncertainty via a *generalised Polynomial Chaos* (gPC) theory and is implemented by differential dynamic programming. The algorithms is able to drive the probabilistic evolution of nonlinear systems with stochastic model parameters. Key features of the system include fast convergence and scalability which are both critical for high-dimensional problems. The aforementioned efficiency traits originate from linearised gPC representations in discrete time via variational integrators. The method was compared against standard numerical optimisation methods and was shown superior in a series of tasks. Even though the work focuses solely on the properties of mechanical systems, it is a good reference for optimisation-based robot motion planners.

Mao *et al.* focused their work on *Real-time Adaptive Motion Planning* (RAMP) in [68]. The proposed algorithm merges approaches of planning with task constraints and planning in dynamic environments without task constraints by continuously maintaining and improving a set of trajectories meeting the requirement of either planning types. This allows the executing framework to seamlessly switch between these sets at any time, resolving conflicts between task and collision avoidance constraints in an on-the-fly fashion. While this approach may not be desirable for all types of tasks, since task-constraints may be lifted for brief period during execution, there are areas where this is viable approach. The method was validated in simulation and real hardware in dynamic scenarios.

This section discussed state-of-the-art motion planning methods. The key features of relevant approaches are summarised in Table 2.2. Optimisation-based methods with

Table 2.2. Summary of relevant motion planning methods and their key features.

Name	Type	Space	Focus	Online	Reference
RRT, RRTConnect	Sampling	Joint/Task	Path planning	Yes	[41], [42]
PRM	Sampling	Joint/Task	Path planning	No	[43]
Experience Graph	Search	Joint	Path planning	Yes	[56]
SBPL	Search	Task	Motion planning	Yes	[58]
TEB	Optimisation	Joint/Task	Motion planning	Yes	[15], [63]
CHOMP	Optimisation	Joint	Motion planning	Yes	[47], [48]
DeBaTo	Optimisation	Joint	LfD	No	[60]
STOMP	Optimisation	Joint	Motion planning	Yes	[52]
GPMP2	Optimisation	Joint	Motion planning	Yes	[65]
TrajOpt	Optimisation	Joint	Motion planning	Yes	[53]
RAMP	Optimisation	Task	Reactive planning	Yes	[68]

online performance are the main interest of this thesis as they natively allow for an initial trajectory to be provided. Some planners operate only in joint space but in most cases this does not disallow formulating cost functions in task space. Achieving the opposite however is hard as task-space planners typically rely on controllers for joint-space execution. In conclusion, the best-fitting planners are TEB, CHOMP, STOMP, GPMP2 and TrajOpt.

Competitions and individual research [69]–[71] both emphasised the importance of a standard set of benchmarks for robotic manipulation. Unfortunately there is not yet a single, well-established benchmark as the adoption of them may be time consuming depending on one's choice of software framework. This thesis utilises a subset of metrics used in the aforementioned benchmarks as well as taking inspiration from experiments done in state-of-the-art literature. The choice of motion planning software framework is MoveIt [72] while the other popular alternative is OpenRave [73]. The choice was made on the former due to the wide availability of ROS-based robots in both research and industrial environments.

2.5 Robot platforms

The *TIAGo* robot depicted in Figure 2.1 is produced by *PAL Robotics*, Barcelona. It is a mobile manipulator robot platform with a two-wheel differential mobile base, pan-tilt head, an elevating column for torso and a 7 *Degrees of Freedom* (DoF) arm with a modular system that supports swapping out *end-effectors*. This robot was designed to solve tasks in *Ambient Assisted Living* and small scale manufacturing and warehousing. In

the following chapters *TIAGo* is used for mobile-base path planning as well as manipulation, using a parallel gripper. The largest available kinematic chain for manipulation is 8 DoF making this robot very capable. More information and simulation model is available online [2].

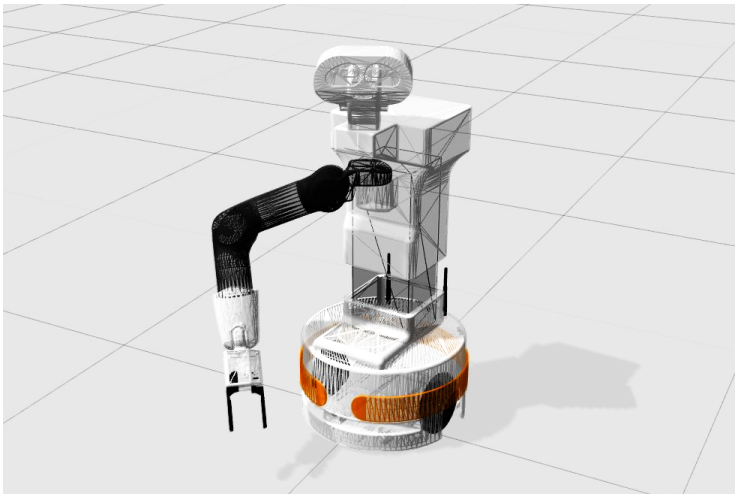


Figure 2.1. The *TIAGo* robot

The *Rob@work3* robot shown in Figure 2.2 was designed for intralogistics tasks and is a product of *Fraunhofer IPA*, Stuttgart. It is a mobile manipulator robot platform with a four-wheel, omnidirectional mobile base and a 6 *Degrees of Freedom* (DoF), *UR10* robot arm that supports swapping out *end-effectors*. This robot focuses on logistics tasks and features a powerful mobile base with a loading area. In the following chapters *Rob@work3* is used for manipulation motion planning using a suction- and parallel gripper. The largest available kinematic chain for manipulation is 6 DoF which is sufficient for common tasks in the warehousing domain. More information and simulation model is available online [3].

[2] http://wiki.ros.org/Robots/TIAGo
[3] https://www.care-o-bot.de/en/rob-work.html

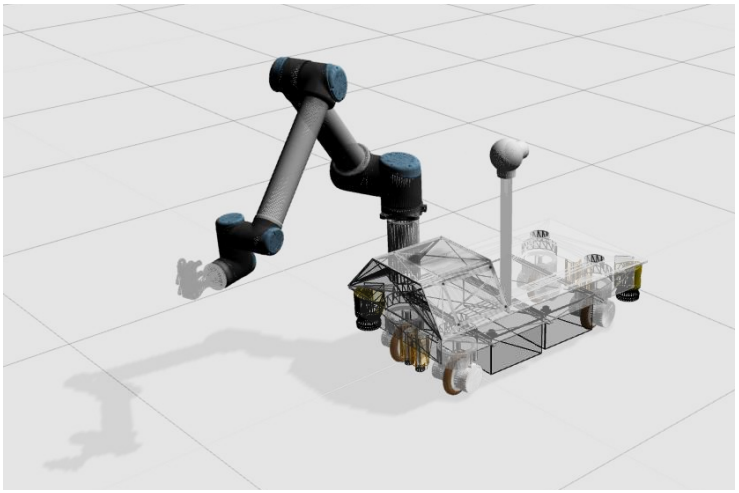


Figure 2.2. The *Rob@work3* robot

2.6 Summary

This chapter introduced trajectory representation and kinematics methods in Section 2.2, presented relevant literature regarding *Learning from Demonstration* (LfD) in Section 2.3 and motion planning methods in Section 2.4. The presented motion planning methods vary in their capability of supporting different trajectory representations and learning approaches.

Finally, Section 2.5 presented the robot platforms used in this thesis.

Chapter 3

Guided Stochastic Optimisation for Motion Planning

MOTION planning for manipulation tasks in human environments is a challenging problem. Robots employed in *Ambient Assisted Living* (AAL) and warehouse environments are beginning to face increasing complexity in both task and environmental settings. The new generation of collaborative robots often supports compliant behaviors and leverages *Learning from Demonstration* (LfD) approaches to allow adapting to new tasks, task instances or environments without specialised tooling and processes. In fact, LfD offers a solution for manipulation problems by using prior demonstrations without explicitly modeling the environments and systems. This alone, however, is not enough to handle changing parameters of a task instance (e.g. picking from a different position on the shelf or opening a different set of drawers) nor capable of collision-free planning. In order to guarantee a robust solution, the experiments in this chapter are done without tweaking internal parameters.

Stochastic Trajectory Optimization for Motion Planning (STOMP) is one of the most popular motion planning frameworks. It plans a trajectory by taking an initial guess and iteratively adapts the trajectory, trying to avoid obstacles and self-collisions. However, choosing a bad initial guess can cause unnecessarily complex plans or failed planning attempts. As a consequence, the optimisation process may be subject to increased planning time and local minima. This chapter presents a novel motion planning approach which introduces *Dynamical Movement Primitive* (DMP)-based LfD for task-informed trajectory generation in STOMP. The generated trajectories, called *guides*, reproduce learned motions for the initialisation of trajectory optimisation. Therefore, it increases planning capability beyond going from A to B, through learning from

demonstration while maintaining the benefits, e.g., optimal collision and joint limit avoidance. The new method, coined *Guided Stochastic Optimization for Motion Planning* (GSTOMP) will be shown in extensive experiments on two manipulation systems to be comparable with state-of-the-art motion planners in various tasks and task instances such as training on one drawer and then opening others. Additional experiments will prove its capability to generate task-specific trajectories from demonstrations alone, without task-specific modeling. Finally, skill transfer is performed between the two manipulation systems as well as a study is presented on varying performance due to *Degrees of Freedom* (DoF) changes.

The following paragraphs relate the presented approach to existing methods and highlight necessary modifications in an overview. A summary of contributions is given before they will be explained in detail in the rest of the chapter. Conceptually similar to *sparse roadmap spanners*' experience graphs [55], GSTOMP uses DMPs to encode such information from a single demonstration. Roadmap spanners focus on high degree of freedom, highly constrained systems such as bipedal humanoid robots and works with experiences recorded in joint space while GSTOMP focuses on arm motion planning by means of pre-recorded Cartesian-space trajectories and stochastic optimisation. Motivated by similar challenges, *Combined Learning from demonstration And Motion Planning* (CLAMP) [62] integrates prior knowledge in the form of *Probabilistic Motion Primitives* (ProMPs) within the *Covariant Hamiltonian Optimization for Motion Planning* (CHOMP) framework. They they adopt a probabilistic approach as GSTOMP to motion representation in joint space however due to the adopted probabilistic approach, they require several different demonstrations of the task. Kyrarini *et al.* [28] also employed kinesthetic teaching to acquire demonstrations for training a *Gaussian Mixture Model*s (GMMs) from several human demonstrations. Unfortunately, the experiments they presented posed no variation to the task being solved to show that it can generalise well. In addition, the approach is a collection of preexisting tools instrumented to solve that specific task, adding small value to the state-of-the-art. In comparison, the approach proposed in this chapter requires only a single demonstration for learning a task. Moreover, it shows that it can generalise well solving multiple task instances of the same task and it can even transfer knowledge between different systems.

Although this chapter does not address the methods of capturing demonstrations and grasp planning, they form an important role in tackling the manipulation problem. Since the GSTOMP framework uses Cartesian-space motions to represent manipulation

skills, many existing techniques can be leveraged to acquire new skills. Kinesthetic teaching is used in the experiments of this chapter to acquire demonstrations, but the same technique can be used to learn from other planners, motion capture systems or even video analysis.

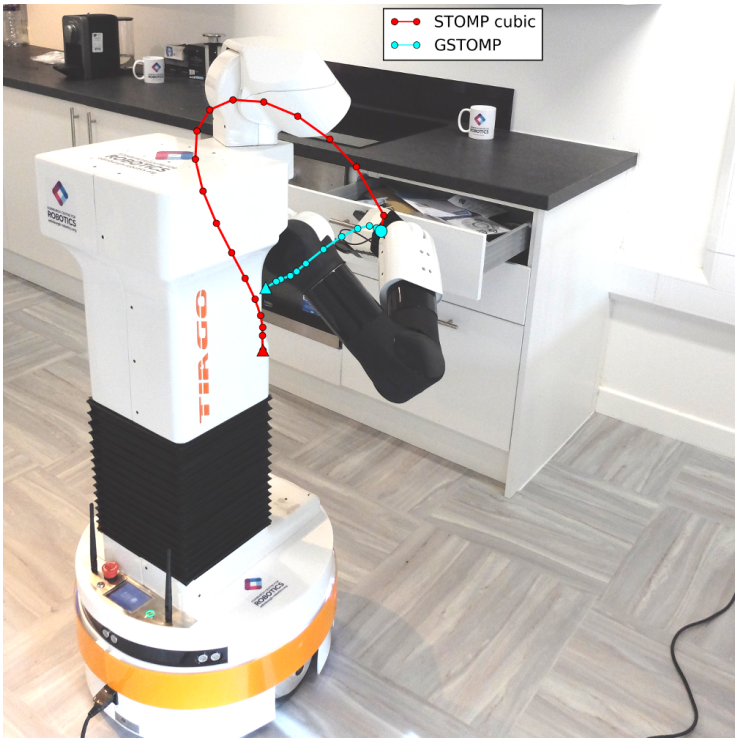


Figure 3.1. Planning for opening a drawer: STOMP cubic and GSTOMP trajectories visualised

STOMP, introduced by Kalakrishnan *et al.* in [52], has been a successful and popular approach to solving the motion planning problem for robot manipulation. Being as versatile as stochastic optimisation, STOMP lacks the ability to handle tasks more complex than *A-to-B* planning and may also suffer from local minima depending on the initial guess (or *seed trajectory* in STOMP terminology). These factors may degrade performance of both speed and success of planning. It would be ideal to start from trajectories that already reach the goal from the start state such that STOMP only needs to make small adjustments (if any) to avoid collisions and joint limits. Additionally, since STOMP works by optimising trajectories in joint-space, there is no guarantee that the optimisation keeps Cartesian-space profile of initial trajectories.

To achieve this, a novel algorithm termed GSTOMP is proposed, which introduces the use of task-informed trajectory initialisation and a new cost function to the optimisation. The importance of the trajectory initialisation for STOMP was first observed

in the work of Schulman *et al.* in [54]. They mentioned that trajectory optimisation methods may require multiple initialisations or task-informed initialisations to improve their success rate. This work focuses on the latter by introducing *guide trajectories* encoded through DMP as *initial trajectories* for STOMP. The new cost function is used to penalise moving away from the *guide trajectory* (i.e., the one generated by DMP as explained in Subsection 3.1.2) in Cartesian-space while still providing obstacle and joint limit avoidance. Experimental results show that introducing a LfD initialisation method, increases the success rate of the planner and allows specific tasks to be successfully completed. This chapter is largely based on [16]. The main contributions are as follows:

1. Introduce a new initialisation strategy which is able to bring task-specific constraints without explicit, task-specific mathematical formulation for STOMP.
2. Propose a novel cost function extending the joint-space optimisation of STOMP with Cartesian-space properties.
3. Extensive experiments on two real manipulators show GSTOMP achieves similar or better results than the state-of-the-art in three different planning scenarios.
4. Demonstrate the capability of GSTOMP for skill transfer between different robots.
5. Show that GSTOMP is more robust to DoF changes in the manipulator system compared to the state-of-the-art.

The remainder of this chapter is structured as follows. Section 3.1 introduces theoretical background for the components of the new approach. Evaluation and comparison with other methods are presented in Section 3.2 while Section 3.3 discusses some of the in-depth details of this method. Finally the chapter is concluded with Section 3.4.

3.1 GSTOMP

GSTOMP combines the optimisation framework of STOMP with a task-informed trajectory generator implemented using Cartesian DMPs and a trajectory distance measure based on *Dynamic Time Warping* (DTW). Figure 3.2 depicts the high-level architecture and draws references to the contributions list in the introduction of this chapter. When a new plan is requested, the trajectory generator uses the kinematic model of the robot,

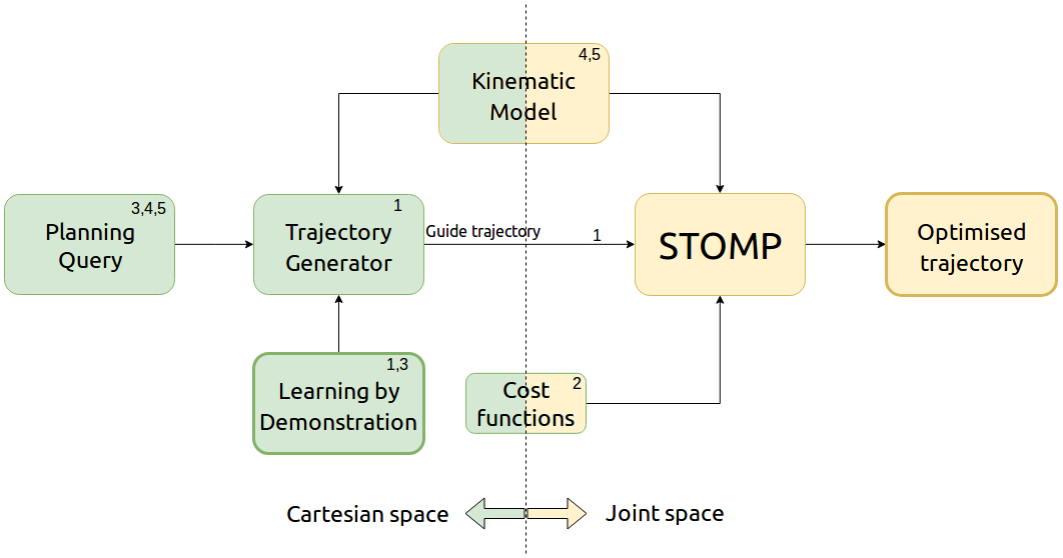


Figure 3.2. The architecture of GSTOMP. Colours denote the space they operate in: yellow marks joint space while green is for Cartesian space. Numbers reference items of the contribution list: 1. Initialisation strategy 2. Extend STOMP with Cartesian costs 3. Extensive experiments with state-of-the-art 4. Skill-transfer 5. Robustness analysis to DoF changes

prior motion information learned from a demonstration corresponding to the desired task, along with the start and desired end position. The generated trajectory is then used to initialise STOMP to perform optimisation using the robot's kinematic model and a set of user defined cost functions. The outcome of this process is an optimised trajectory that can be sent to the robot's controllers in order to perform the requested task.

The three steps of acquiring a solution within GSTOMP is shown in Figure 3.3 through an example. A demonstration was recorded reaching for a target marked by the blue line. The starting pose of the trajectory is denoted by the circle on the right side of the plot, while the ending pose of the trajectory is shown as a triangle on the left side. The new target is on the same surface but located to the right from the previous one. When a plan for this new target is requested, a guide trajectory is generated as shown by the purple line. Note how the shape is preserved in the *guide trajectory* and how the optimisation modifies it, creating a robot trajectory marked by the red line. Circle and triangle markers mark the start and end states respectively.

Since the DMP-generated *guide trajectory* is in Cartesian space and STOMP operates in joint space, a one-time Cartesian to joint space conversion is necessary. *TRACLabs Inverse Kinematics Solver* (TRAC-IK) [21] is employed in GSTOMP as it has been shown to outperform the popular KDL solver. The use of *Forward Kinematics* (FK) on

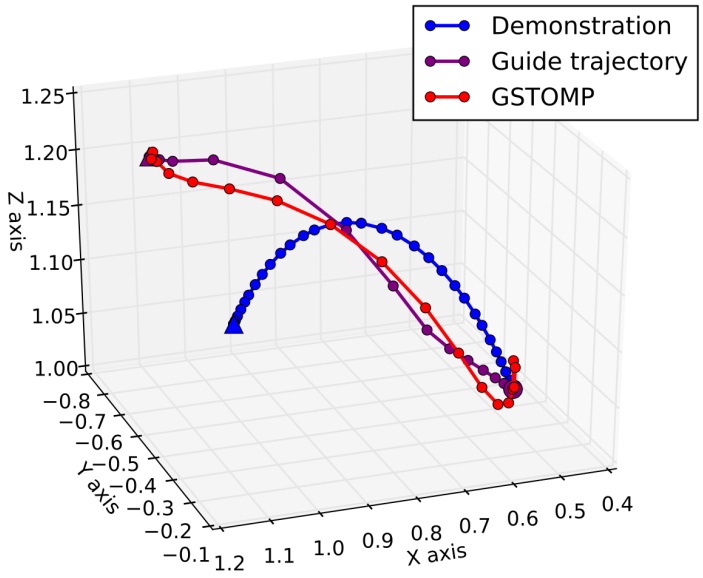


Figure 3.3. Cartesian-space trajectories from three stages of GSTOMP. Circles and triangles mark where trajectories start and end respectively.

the other hand does not incur significant computational costs and is unequivocal operation hence why it was chosen to be used for the trajectory similarity function. More details on this subject were discussed in Section 2.2. The rest of this section briefly introduces each component.

3.1.1 Interpolation-based trajectory generation

Simple, non-task-informed trajectory generation methods can also be used for initialising optimisation-based planners.

Given an initial state $\mathbf{\Phi}_0$ and a goal state $\mathbf{\Phi}_g$ a joint-space trajectory is defined as $\Theta = \{\mathbf{\Phi}_0, \ldots, \mathbf{\Phi}_i, \ldots, \mathbf{\Phi}_{N-2}, \mathbf{\Phi}_g\}$ with a length of N.

In the following definitions multiplication with scalars and addition of trajectories is used in an element-wise fashion.

Linear Interpolation

A linearly interpolated trajectory can be computed as a serie of weighted sums of the initial and the goal states. An element of such a trajectory can be expressed as

$$\mathbf{\Phi}_i = \frac{i}{N} \cdot \mathbf{\Phi}_g + \frac{N-i}{N} \cdot \mathbf{\Phi}_0 \tag{3.1}$$

The benefit of this interpolation method is that it provides an easy-to-follow, constant velocity trajectory.

Cubic Interpolation

The intermediate point can also be estimated using cubic spline interpolation.

$$c_0 = \mathbf{\Phi}_0 \tag{3.2}$$

$$c_2 = \frac{3}{(N-1)^2} \cdot (\mathbf{\Phi}_g - \mathbf{\Phi}_0) \tag{3.3}$$

$$c_3 = \frac{-2}{(N-1)^3} \cdot (\mathbf{\Phi}_g - \mathbf{\Phi}_0) \tag{3.4}$$

$$\mathbf{\Phi}_i = c_0 + c_2 \cdot i^2 + c_3 \cdot i^3 \tag{3.5}$$

This interpolation guarantees continuity at the velocity level in the trajectory. During the experiments this interpolation method was found to explore a larger part of the manipulator workspace with a smoother trajectory.

Minimal Control Cost Interpolation

Algorithm 3.1 describes an efficient method of generating a minimum control cost trajectory. In scope of this work control cost is defined by the sum of joint-level accelerations but the method is parametrisable through the definition of $R \in \mathbb{R}^{N \times N}$, a positive semi-definite matrix representing control costs.

Algorithm 3.1 Minimum control cost trajectory interpolation

Require: $\mathbf{\Phi}_0, \mathbf{\Phi}_g, N, \Delta_t$
Ensure: Θ (minimum cost trajectory from $\mathbf{\Phi}_0$ to $\mathbf{\Phi}_g$)

1: **for** $j \leftarrow 0$ to $length(\mathbf{\Phi}_0)$ **do**
2: $\quad lincost = 2 \cdot (\mathbf{\Phi}_0(j) \cdot I_j \cdot R_1 + \mathbf{\Phi}_g(j) \cdot I_j \cdot R_N)$
3: $\quad \Theta(j) = -0.5 \cdot R^{-1} \cdot lincost^T$
4: **return** Θ

The initial and goal states are defined in joint space and denoted $\mathbf{\Phi}_0$ and $\mathbf{\Phi}_g$. The number of waypoints is a parameter denoted by N, while Δ_t stands for the constant time difference between these waypoints. The algorithm uses j, joint index and M is the

length of the used Finite Central Differentiation rule, set to 5 in this work as it captures acceleration, while $I_j = [1, \dots, 1] \in \mathbb{R}^{(1\times M)}$ vector of values. $R_1, R_N \in \mathbb{R}^{M\times N}$ are the block of R corresponding to the first and the last trajectory points respectively. $R^{-1} \in \mathbb{R}^{N\times N}$ the inverse of R.

R is defined in Equation 3.6 using the Finite Central Differentiation rule for acceleration in Equation 3.7. The Δ_t parameter was dropped for brevity.

$$R(\Delta_t) = \begin{pmatrix} FCD(\Delta_t) & 0 & \dots & 0 \\ 0 & FCD(\Delta_t) & \dots & 0 \\ \vdots & \vdots & \ddots & \vdots \\ 0 & 0 & \dots & FCD(\Delta_t) \end{pmatrix} \tag{3.6}$$

$$FCD(\Delta_t) = \frac{1}{\Delta_t^2}\begin{bmatrix} \frac{-1}{12} & \frac{16}{12} & \frac{-30}{12} & \frac{16}{12} & \frac{-1}{12} \end{bmatrix} \tag{3.7}$$

3.1.2 Task-informed trajectory generator

A popular approach for generating trajectories from demonstrations in a flexible way is *Dynamical Movement Primitive*s introduced by Ijspeert *et al.* in [29]. Specifically for this case Cartesian Dynamical Movement Primitives [30] are used, which can be defined as a combination of position and rotation components $(\text{DMP}_{\text{pos}}, \text{DMP}_{\text{rot}})$. The necessity of separating the position component from the rotational part lies in the fact that rotations cannot be reliably represented by $\mathbb{R}^3$ but have to be either in unit quaternion or rotation matrix form. For a fully comprehensive description of orientation DMPs the reader is kindly referred to the work of Ude *et al.* in [31].

The position DMP component is described by the damped spring model as shown in the following equation:

$$\tau\ddot{y} = \alpha_z(\beta_z(y_g - y) - \dot{y}) + f(x) \tag{3.8}$$

where y is the current position, τ is a scaling factor for time, y_g is the goal position. α_z and β_z are scaling terms and f is a forcing term. The forcing term f is defined as:

$$f(x) = \frac{\sum_{i=1}^{B} \mathbf{\Psi_i}(\mathbf{x})\mathbf{w_i}}{\sum_{i=1}^{B} \mathbf{\Psi_i}(\mathbf{x})} x(y_g - y_0) \tag{3.9}$$

where y_0 stands for the initial state while y_g is the goal state, B is the number of basis functions, $\mathbf{w_i}$ is weighting for a given basis function Ψ_i. While B is user defined, $\mathbf{w}$ is

learned from demonstration. The time element in this formula is represented by its own first order linear dynamics such that

$$\tau \dot{x} = -\alpha_x x \tag{3.10}$$

where α_x is a constant. $\mathbf{\Psi_i(x)}$ is exponential basis functions:

$$\mathbf{\Psi_i(x)} = \exp\left(-\frac{\mathbf{(x - c_i)^2}}{\mathbf{2\sigma_i^2}}\right) \tag{3.11}$$

where $\sigma_\mathbf{i}$ and $\mathbf{c_i}$ define the width and centers of the basis functions, respectively. For instance, if f is defined constant 0, the y term converges to the goal position y_g following a second order linear system dynamics defined by the parameters α_z and β_z. For a thorough discussion of the characteristics, implementation and training of DMPs the work of Ijspeert *et al.* in [29] is highly recommended.

Regarding the rotation DMP component, it is known that no minimal representation of orientation exists in $\mathbb{R}^3$ that contains no singularities and that its differentiation is numerically stable. To solve this problem, Ude *et al.* in [31] proposed solutions using rotation matrices or unit quaternions. GSTOMP employs quaternion DMPs in its Cartesian DMP implementation. However, quaternion is a non-minimal representation as $\mathbf{q} \in \mathbb{R}^\mathbf{4}$. Consequently, this DMP formulation cannot be directly used as it assumes independent numerical values for each degree of freedom.

As Kramberger *et al.* have shown in [30], Equation 3.8 can be rewritten in quaternion format as two equations, one to cover the original acceleration-space damped spring equation, and one in velocity space as

$$\tau \dot{\eta} = \alpha_z(\beta_z 2log(\mathbf{q_g} * \bar{\mathbf{q}}) - \eta) + \mathbf{f_o(x)} \tag{3.12}$$

$$\tau \dot{\mathbf{q}} = \frac{1}{2}\eta * \mathbf{q} \tag{3.13}$$

where $\eta \in \mathbb{R}^3$ is the angular velocity vector with individual velocity components along the x, y and z axes, $*$ denotes the quaternion multiplication and f_o is a learned forcing term, the equivalent of f but for orientations. Note how the goal term changed to a quaternion difference which is defined by multiplication by the conjugate. The logarithm of this difference returns an angular velocity vector, where the logarithm of a

quaternion such that $\log : \mathbb{S}^3 \rightarrow \mathbb{R}^3$ is defined below:

$$\log(\mathbf{q}) = \log(\mathbf{v} + \mathbf{u}) = \begin{cases} \arccos(v)\frac{\mathbf{u}}{\|\mathbf{u}\|}, & \|\mathbf{u}\| \neq \mathbf{0} \\ \mathbf{0} \in \mathbb{R}^{\mathbf{3}}, & \text{otherwise} \end{cases} \tag{3.14}$$

where $\mathbf{q}$ can be deconstructed into a scalar part (v) and a vector part ($\mathbf{u}$).

The DMP function approximator requires accelerations therefore the desired trajectory $\mathbf{q^{des}}$ is differentiated.

$$\eta_t^{\text{des}} = \frac{2 \times \log(\mathbf{q_t^{des}} * \bar{\mathbf{q}}_{\mathbf{t-\Delta t}}^{\text{des}})}{\Delta t} \tag{3.15}$$

Subsequently, velocities are differentiated to acquire accelerations as follows:

$$\dot{\eta}_t^{\text{des}} = \frac{\eta_t^{\text{des}} - \eta_{t-\Delta t}^{\text{des}}}{\Delta t}. \tag{3.16}$$

The first elements are initialised to $\mathbf{0}$ as in the beginning the robot is not moving: $\eta_{des}(0) = \mathbf{0} \in \mathbb{R}^{\mathbf{3}}$, $\dot{\eta}_{des}(0) = \mathbf{0} \in \mathbb{R}^{\mathbf{3}}$.

Finally, the target values to fit a function to can be summarised as

$$\mathrm{f}_{\text{target}} = \dot{\eta} - \alpha_{\mathbf{z}}(\beta_{\mathbf{z}}\mathbf{2}\log(\mathbf{q_o} * \bar{\mathbf{q}}_{\mathbf{j}}) - \eta_{\mathbf{j}}). \tag{3.17}$$

To generate a new trajectory Equation 3.12 is integrated. For integrating Equation 3.13 the inverse of the log operator is applied. The exponential of an angular velocity vector is such that $exp : \mathbb{R}^3 \rightarrow \mathbb{S}^3$ defined below:

$$exp(\mathbf{r}) = \begin{cases} \cos(\|\mathbf{r}\|) + \sin(\|\mathbf{r}\|)\frac{\mathbf{r}}{\|\mathbf{r}\|}, & \|\mathbf{r}\| \neq \mathbf{0} \\ \mathbf{0} \in \mathbb{S}^{\mathbf{3}}, & \text{otherwise} \end{cases} \tag{3.18}$$

3.1.3 Stochastic Optimisation over guide trajectories

STOMP uses an initial guess for the optimisation process. This has to be a joint-space trajectory driving the arm from start state to goal state. Three different initialisation strategies are available currently: linear interpolation, cubic interpolation and a minimal control cost trajectory. In the algorithm description this initial trajectory is denoted as $\mathbf{X}_{\text{guide}}$. One of the main contributions is to allow trajectories generated by Cartesian DMPs to be used as $\mathbf{X}_{\text{guide}}$ after being converted to joint space.

Algorithm 3.2 presents the proposed GSTOMP approach, building on STOMP introduced by Kalakrishnan *et al.* in [52]. A is a finite differencing matrix which produces accelerations $\ddot{\Theta}$ when multiplied by the state vector Θ. M is a smoothing matrix used in the update step of the optimisation, $\mathcal{N}$ is a set of normal distributions to sample noise from, while S and P denote the per-timestep cost and probabilities of each noisy trajectory respectively. $\overline{\Theta}_{k,i}$ denotes the i-th state in $\overline{\Theta}_k$ and Θ_i denotes the i-th state in Θ, dof stands for the number of DoF of the kinematic chain. Further specifics on the STOMP algorithm are described in [52].

Algorithm 3.2 THE GSTOMP ALGORITHM

Given:
- Start ($\mathbf{\Phi}_0$) and goal ($\mathbf{\Phi}_N$) states, $\mathbf{\Phi}_i \in \mathbb{R}^{\text{dof}}$
- A discretised trajectory $\mathbf{X}_{\text{guide}} \in \mathbb{R}^{\text{dof} \times N}$
- A state-dependent cost function $q(\mathbf{\Phi}_i)$

Precompute:
- A = finite difference matrix
- R^{-1} = $(A^T A)^{-1}$
- M = R^{-1}, with each column scaled such that the maximum element is $(1/N)$
- Let $\Theta = \mathbf{X}_{\text{guide}}$

Repeat until convergence of trajectory cost $Q(\Theta, \mathbf{X}_{\text{guide}})$:

1) Create K noisy trajectories, $\overline{\Theta}_1 .... \overline{\Theta}_K$ with parameters $\Theta + e_k$ where $e_k = \mathcal{N}(0, R^{-1})$

2) For $k = 1...K, i = 1...(N-1)$ compute:

   a) $S(\overline{\Theta}_{k,i})$ = $q(\overline{\Theta}_{k,i})$

   b) $P(\overline{\Theta}_{k,i})$ = $\frac{e^{-\frac{1}{\lambda} S(\overline{\Theta}_{k,i})}}{\sum_{l=1}^{K} [e^{-\frac{1}{\lambda} S(\overline{\Theta}_{l,i})}]}$

3) For $i = 1...(N-1)$, compute: $[\delta\overline{\Theta}]_i$ = $\sum_{k=1}^{K} P(\Theta_{k,i})[e_k]_i$

4) Compute $\delta\Theta$ = M$\delta\overline{\Theta}$

5) Update $\Theta \leftarrow \Theta + \delta\Theta$

6) Compute $Q(\Theta, \mathbf{X}_{\text{guide}})$ according to Equation 3.24

3.1.4 Cartesian trajectory cost function

Providing a *guide trajectory* that looks like a good solution is not a guarantee for the trajectory to be maintained during the optimisation process. To motivate exploration close to the *guide trajectory*, a cost function on the similarity between the *guide* and optimised trajectories is introduced. *Dynamic Time Warping* (DTW) originated from natural language processing is a robust distance measure for two sets of time-series data. It was first defined by Sakoe *et al.* in [74]. In contrast to existing approaches such

as the one defined in [51], the DTW implementation of this chapter follows the original equations with a distance function on the Cartesian trajectory. This distance metric is defined by a weighted sum of the Euclidean distance of the position components and the quaternion logarithm error (used in Equation 3.12) of the orientation components. The distance function used in the DTW algorithm is defined as

$$d(i,j) = \mathbf{w}_{pos}\|\mathbf{p_i}, \mathbf{p_j}\| + \mathbf{w_{ori}2}\|\mathbf{log}(\mathbf{q_i} * \bar{\mathbf{q}}_\mathbf{j})\| \tag{3.19}$$

where i, j are indices, each traversing a different trajectory, *e.g.* $\mathbf{p_i}$ and $\mathbf{q_i}$ denote the position and orientation respectively of the i-th element in the first trajectory, j works similarly on the second trajectory. During the experiments of this work, the proportion $\frac{\mathbf{w}_{pos}}{\mathbf{w}_{ori}} = 4$ was found to be ideal. This compensates for the numerical differences of the two distance metrics. For brevity, the Cartesian-space DTW implementation is referred to as C-DTW $\in \mathbf{X} \times \mathbf{X} \rightarrow \mathbb{R}$ and is defined as the sum of element-wise time-warped distances:

$$\text{C-DTW}(\mathbf{X}_A, \mathbf{X}_B) = \Sigma_{i \in Ind(\mathbf{X}_A), j \in Ind(\mathbf{X}_B)} d(i,j) \tag{3.20}$$

where $Ind(\mathbf{X})$ denotes the indices of trajectory $\mathbf{X}$. Note that thanks to this formulation, the length of $\mathbf{X}_A$ and $\mathbf{X}_B$ do not need to be the same as DTW will always pick closest corresponding points from the two trajectories when computing the distance.

3.1.5 GSTOMP cost function

The composed cost function of GSTOMP takes collision checks, collision proximity and similarity to initial trajectory $\mathbf{X}_{\text{guide}}$ into account by calculating a weighted sum of these components.

A collision checker function which assigns high values to states with collisions is defined as follows,

$$\xi(\Theta_i) = \begin{cases} \text{collision\_penalty}, & \text{collisionWithWorld}(\Theta_i) \text{ or collisionWithSelf}(\Theta_i) \\ 0, & \text{otherwise} \end{cases} \tag{3.21}$$

where the boolean collision values are computed by collisionWithWorld(Θ_i) and collisionWithSelf(Θ_i) on a single node Θ_i of the trajectory for collisions with the envi-

ronment and the robot body, respectively. When used with an entire trajectory Θ as a parameter, $\xi \in \Theta \rightarrow \mathbb{R}^N$, the function returns a vector of collision flags.

The following function defines a single collision gradient computation.

$$\upsilon(\Theta_i) = \begin{cases} 0, & \text{collisionDistance}(\Theta_i) > d^{\text{L}} \\ \frac{d^{\text{L}} - \text{collisionDistance}(\Theta_i)}{d^{\text{L}}}, & 0 < \text{collisionDistance}(\Theta_i) < d^{\text{L}} \\ \text{collision\_penalty}, & \text{collisionDistance}(\Theta_i) < 0 \end{cases} \tag{3.22}$$

with collisionDistance(Θ_i) denoting the computation of the distance of any robot body part to the environment and itself, this is provided by the Flexible Collision Library presented by [75]. For the experiments conducted in this chapter, the value of d^{L} was set to 0.2m. The function which explores neighbouring states and assigns a cost based on the proximity of states with collisions can now be defined as

$$\Upsilon(\Theta) = \Sigma_{i=1}^{N-1}(\upsilon(\Theta_i) + \Sigma_{j=1}^{L}\upsilon(\mathscr{I}(\Theta_i, \Theta_{i+1}, \frac{j}{L}))) \tag{3.23}$$

where $\mathscr{I} \in \Theta \times \Theta \times \mathbb{R} \rightarrow \Theta$ is a linear interpolation method with a scalar parameter $s \in [0, 1]$ determining the phase of transition between the first and the second trajectory argument, N is the number of nodes in the trajectory and L is the number of intermediate points to check which was set to 5 in this chapter's experiments.

Finally, the GSTOMP cost function is defined as $Q(\Theta, \mathbf{X}_{\text{guide}})$, reflecting that this is a STOMP cost function employed in Algorithm 3.2 using both the working trajectory (Θ) and the initial trajectory $(\mathbf{X}_{\text{guide}})$. In the experiments of this chapter, collision avoidance is prioritised first, *guide* following second, and staying far from colliding states last.

$$Q(\Theta, \mathbf{X}_{\text{guide}}) = w_c\xi(\Theta) + w_{cg}\Upsilon(\Theta) + w_{dtw}\text{C-DTW}C_{FK}(\Theta), \mathbf{X}_{\text{guide}}) + w_{cc}\frac{1}{2}\Theta^T R\Theta \tag{3.24}$$

where ξ is the collision checker function from Equation 3.21, Υ is a collision gradient function defined in Equation 3.23, C-DTW is the Cartesian trajectory distance defined in Equation 3.20, $C_{FK} \in \Theta \rightarrow \mathbf{X}$ is a function converting joint-space trajectories to task-space trajectories using FK and $\Theta^T R\Theta$ is the control cost in the STOMP framework. The corresponding weighting factors are denoted by w_c, w_{cg}, w_{dtw} and w_{cc} respectively.

Supporting a large variety of tasks with a method that may require tuning param-

eters between tasks is a near impossible challenge. Based on the experience of this work, STOMP and GSTOMP only require tuning a single time per collection of cost functions. Once the weighting is adjusted such that it balances all components to a similar order of magnitude - taking preferences into account - the weights do not need further adjustment.

3.2 Experiments

After introducing the GSTOMP approach in detail, this section focuses on common manipulation tasks around a home or a warehouse. The following tasks were studied:

- picking and placing an item from and to various positions on a set of shelves as shown in Figure 3.6a, as it was also done at the Amazon Picking Challenge [70],
- opening a drawer shown in Figure 3.6b,
- opening a cupboard shown in Figure 3.6c, as these were also done by Jain *et al.* in [76].

The shelf-picking experiment uses the 6 DoF mobile manipulator robot *Rob@work3* introduced Figure 2.2, designed to perform light industrial tasks and features a vacuum gripper. The drawer and cupboard opening experiments were conducted using the 8 DoF mobile manipulator robot, *TIAGo* introduced in Figure 2.1. It was designed for AAL and uses a parallel gripper. In addition to the aforementioned scenarios, GSTOMP was evaluated in two special studies. In the first case, the DoFs of the robot were reduced, showing the benefit of the *guide* trajectory making GSTOMP more robust on different platforms. In the second case, a trajectory learned by one robot was used to perform tasks using another robot showing skill transfer between platforms.

To evaluate GSTOMP, it was compared with the following state-of-the-art planners and raw Cartesian DMP approaches:

1. *Rapidly-exploring Random Tree Connect* (RRTConnect): a state-of-the-art motion planner from *Open Motion Planning Library* (OMPL)[77]
2. CHOMP: a planner similar to STOMP in nature
3. STOMP linear: STOMP using the linear interpolation initialisation strategy defined in Section 3.1.1

Table 3.1. Metrics used to evaluate each experiment.

Metric	Description
$\mathscr{T}$	Planning time in s
s_r	Success rate in %
$\ddot{\Theta}$	Smoothness in m/s^2

4. STOMP cubic: STOMP using the cubic spline interpolation initialisation strategy introduced in Section 3.1.1

5. STOMP min control: STOMP using a minimal control interpolation initialisation strategy defined in Section 3.1.1

6. STOMP Cartesian line: STOMP using a linear interpolated straight Cartesian line as *guide* strategy

7. Cartesian DMP: A DMP-generated Cartesian *trajectory*. Plans are validated for joint limits and collisions in the same framework without performing further optimisation on it. Although it may be referred to as a planner in the experiments section, it serves as baseline for comparison.

8. GSTOMP: STOMP with a DMP-generated *guide* strategy

The metrics presented in Table 3.1 are used to evaluate speed, success rate and smoothness quality of the trajectories of every planning query. $\mathscr{T}$ is the time planners take between a planning request is sent and the arrival of a response. s_r represents the performance of the planner, it stands for the number of successful responses (collision-free and respecting joint limits) out of 600 planning requests. Similar metrics were discussed in [78]. However, since the motion planners are evaluated in very specific scenarios and the focus is not on shortest path, trajectory length in Cartesian and joint space are not strongly relevant. In these experiments, therefore, the only metric used for quality is Cartesian trajectory smoothness. The smoothness value is a cumulative function of the *end-effector* linear and angular accelerations. The accelerations are approximated using *Second Order Central Difference Approximation*. While the interpretation of other metrics is intuitive, it is important to highlight that in case of the smoothness metric lower values are favoured.

Task training

This subsection provides insights into the practicalities of task training.

Kinesthetic teaching took place on the real robots using the real furniture prior to experiments. The process of this started by putting the robot manipulator into gravity compensation mode so it is freely movable by hand. Following, the robot arm was moved into an assumed grasp pose, for instance, the gripper being around the drawer's handle. As the instructor moved the manipulator to open a drawer, the change in all robot joint positions reported at 200Hz was recorded. The final form of the demonstration was generated by feeding the recording back into the kinematic model and generating the Cartesian-space trajectory of the *end-effector*.

When using a vacuum gripper, perfect grasping can be assumed at the target positions. Using the parallel gripper one has some freedom in rotation around the handles as well as along the length of the handlebar. An effect of this was discovered during the kinesthetic teaching phase: the shape of cupboard opening demonstrations does not follow the curve traveled by the door handle. This is explained by experiencing a higher rate of *end-effector* rotational change in the demonstration trajectory and some travel along the length of the handle as mentioned earlier.

Figure 3.4. Demonstrating cupboard-opening on *TIAGo*

GSTOMP requires a single demonstration per task and the experiments conducted for this work did not highlight any particular sensitivity to properties of demonstrations. One exception is that demonstrations should be one smooth motion, without stopping for longer periods between positions as the reproduction will reflect the same stops. Jerky or abrupt demonstrations will produce similar qualities in the produced plan.

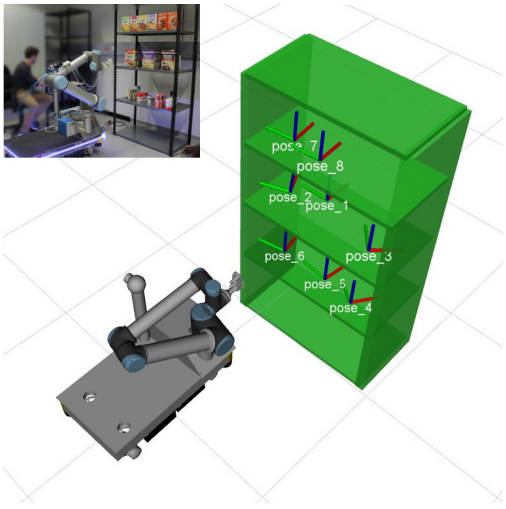
Figure 3.5. Demonstrating a reach motion on *Rob@work3*

Experiments setup

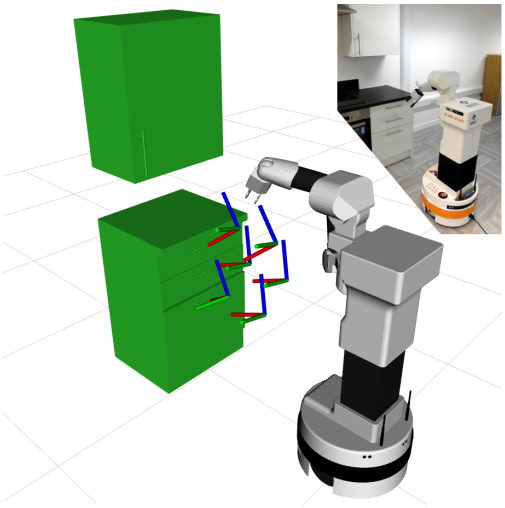
Each environment used in the experiments designed for this chapter was set up with the 3D models of the corresponding robot and furniture available in the respective labs. Planning queries were made against all planners using a set of pre-defined start and end points assuming the same static position of each robot with respect to the furniture. Figure 3.6 depicts these scenarios with the assumed robot positions along with initial *end-effector* conditions and desired Cartesian-space positions denoted by coordinate frames in the figures.

All STOMP variants were set to use 20 samples from a trajectory throughout the following experiments.

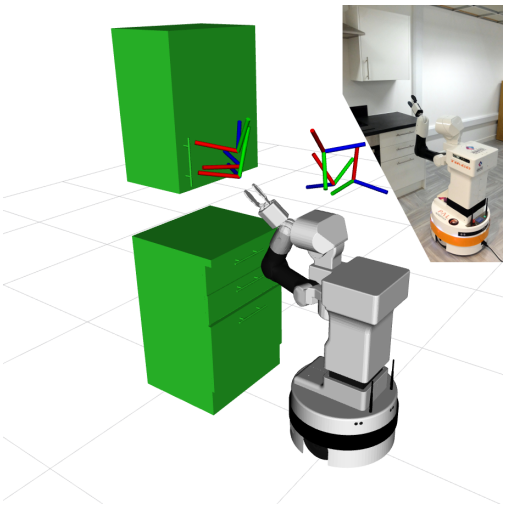
All experiments were conducted using the *MoveIt!* framework [72] on a system with an Intel i7-4710MQ, 2.5GHz CPU, 8GB of RAM and running on Ubuntu 16.04. STOMP and GSTOMP were implemented in C++, while the DMP framework was implemented in Python. Although the experiments and benchmarks of this chapter were carried out on simulated robots, the output of GSTOMP was validated on real robots with the same furniture from the experiments. Since this work does not involve any perception, the robots were started with a scenario having already achieved some sort of grasping in case of the furniture scenarios and for the pick and place example, objects were expected to be at known positions. For validation on *TIAGo* and *Rob@work3*, the *joint_trajectory_controller* (*ros_control* [79]) was used to execute the generated plans. The particularities about the *end-effector* mentioned in Section 3.2 were confirmed by the real robot experiments. Given sufficient stability of objects, the suction gripper



(a) Shelf-picking scenario



(b) Drawer opening scenario



(c) Cupboard opening scenario

Figure 3.6. The scenarios used for verifying GSTOMP. Initial *end-effector* conditions and desired Cartesian-space positions are marked by coordinate frames.

successfully attached objects in the pick experiment while the parallel gripper added some tolerance for the angle of grasping drawer and cupboard handles.

3.2.1 Shelf-picking

This task provides an equal ground for all planners as is a simple *A-to-B* type of planning setup. Different target poses within shelves resemble randomly-placed objects on the three shelves, providing a good set of task instances for evaluation.

This experiment consisted of 100 reaching and 100 retrieving queries for each of the three shelves. Retrieval queries were created by swapping start and end states of

Table 3.2. Parameter values used in all experiments with the *TIAGo* robot

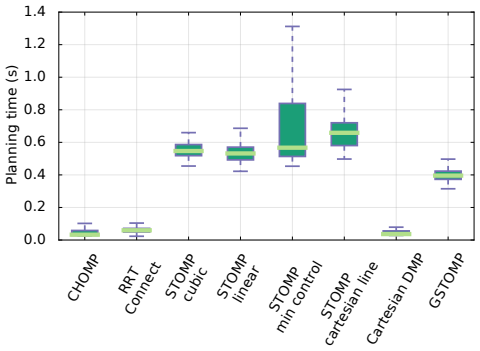
Joint name	Std. dev.	Name	Value
torso joint	0.001	$\mathbf{w}_c$	1.0
arm joint 1	0.001	$\mathbf{w}_{cg}$	1.0
arm joint 2	0.001	$\mathbf{w}_{dtw}$	5.0
arm joint 3	0.01	$\mathbf{w}_{cc}$	0.5
arm joint 4	0.01	K	10
arm joint 5	0.01	$L_{iterations}$	50
arm joint 6	0.01		
arm joint 7	0.01		

Table 3.3. Parameter values used in all experiments with the *Rob@work3* robot

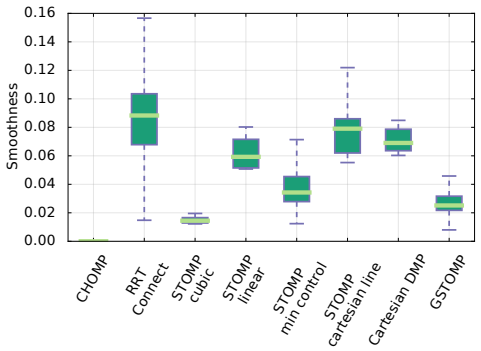
Joint name	Std. dev.	Name	Value
arm joint 1	0.05	$\mathbf{w}_c$	1.0
arm joint 2	0.05	$\mathbf{w}_{cg}$	1.0
arm joint 3	0.05	$\mathbf{w}_{dtw}$	1.0
arm joint 4	0.1	$\mathbf{w}_{cc}$	0.5
arm joint 5	0.1	K	10
arm joint 6	0.1	$L_{iterations}$	50

the reaching queries. A total of 600 queries were made against this scenario using one demonstration for each shelf as shown in Figure 3.6a.

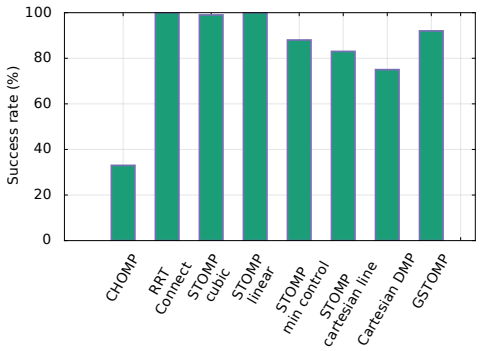
The results are presented in Figure 3.7 and Table 3.4. It shows that RRTConnect, CHOMP and DMP as the fastest methods while GSTOMP performs faster than the other STOMP variants. Regarding smoothness, GSTOMP generates similarly smooth trajectories to other STOMP variants while for this task CHOMP triumphs on this metric. RRTConnect performs this task with high variability, while DMP is worse than the proposed GSTOMP. Since smoothness is computed on the Cartesian path, this proves that GSTOMP manages to stay relatively close to the mostly smooth *guide trajectory*, even improving it as shown by the results. As shown in Figure 3.7c, GSTOMP shows a similar success rate as other STOMP variants and RRTConnect. The pure DMP planner performed worse than GSTOMP, any of the STOMP variants and RRTConnect. This is caused by its total ignorance of the environment that led to collisions. The failure rate of CHOMP due to violating collision constraints is noteworthy, given that even DMP performed better. Figure 3.8 depicts the execution of a GSTOMP trajectory on the real robot.



(a) Planning time, lower values are favoured



(b) Smoothness, lower values are favoured



(c) Success rate, higher values are favoured

Figure 3.7. Shelf-picking scenario experiment results

Figure 3.8. Shelf picking experiment executed on the real *Rob@work3* robot drawn with the *onion skinning effect*

Table 3.4. Shelf-picking scenario: statistics from a total of 600 queries with each planner

Algorithm	Planning time (s)		Smoothness		Success rate
	Median	Std dev	Median	Std dev	
CHOMP	0.043	0.036	**0.000**	0.000	33%
RRTConnect	0.063	0.019	0.092	0.039	**100%**
STOMP cubic	0.558	0.225	0.015	0.002	99%
STOMP linear	0.544	0.159	0.062	0.010	**100%**
STOMP min control	0.592	0.537	0.037	0.018	88%
STOMP Cartesian line	0.667	1.059	0.082	0.022	83%
Cartesian DMP	**0.038**	0.016	0.077	0.015	75%
GSTOMP	0.401	0.107	0.026	0.009	93%

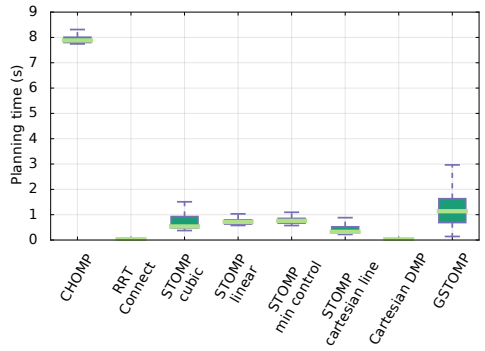
3.2.2 Drawer opening

Similarly to the experiment in Subsection 3.2.1, the drawer opening task is also an *A-to-B* type planning problem with an additional constraint that the planned path should align with the motion of the drawer rails. This attribute was visually validated during the benchmarking process.

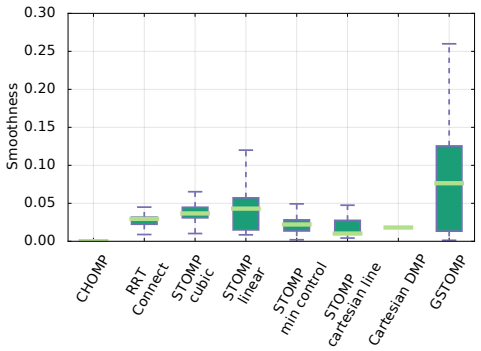
A demonstration was acquired by kinesthetic teaching when opening the top drawer with a real *TIAGo* robot and laboratory furniture. The same demonstration is used to generate *guide trajectories* for the task instances opening the other drawers as well. Start and end locations for the middle and lower drawers were derived from the demonstration but with slight adjustments to accommodate the different grasp angles and positions as shown in Figure 3.6b.

The results of the drawer opening scenario are presented in Figure 3.9 and Table 3.5 from 600 queries with each planner. For visual assessment, one typical drawer-opening trajectory by each planner was drawn in Figure 3.10.

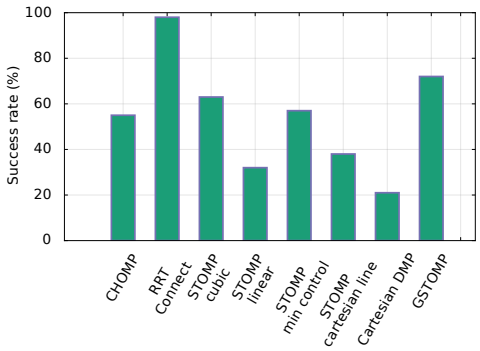
Regarding the planning time the proposed approach is close to most of the other approaches, with exception of CHOMP which performs much worse. It is clear that RRTConnect performed fastest in this scenario. A key factor in this is that RRTConnect explores both from the start and end states at the same time. DMP performed in a similar manner as RRTConnect given the computational simplicity to rollout a motion policy. GSTOMP performed statistically a bit worse than most of the other STOMP variants. On the other hand, CHOMP, performed much slower that any other planner requiring around 8 seconds for a plan. Regarding smoothness CHOMP again provides



(a) Planning time, lower values are favoured



(b) Smoothness, lower values are favoured



(c) Success rate, higher values are favoured

Figure 3.9. Drawer scenario experiment results

the smoothest trajectories. The other planners performed in a comparable manner, with GSTOMP having the most variant performance. RRTConnect was the most successful at finding paths. GSTOMP performed second having a 13% higher success rate than other STOMP variants. This shows the benefit of the *guide trajectories* that helped avoid collisions. An additional benefit to that is that the *guide trajectories* allow the task to be successfully completed. As it can be seen in Figure 3.10 most of the trajectories found by various planners are not straight making them unsuitable for a drawer opening task. CHOMP's' success rate was on par with most of the STOMP approaches. The least performing method was the pure Cartesian DMP. While this method has all the information regarding the task at hand, it fails to find valid trajectories as it ignores the obstacles in the environment and ends up in colliding states. As mentioned in Section 3.2, execution with the cage grasp of the parallel gripper worked because it added rotational tolerance along the handle's shaft.

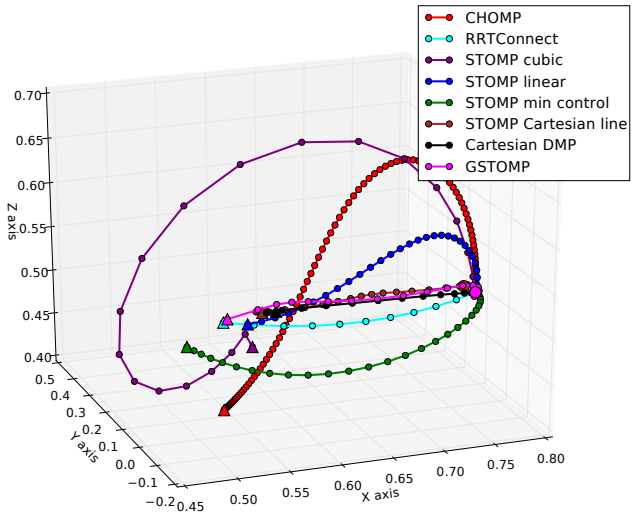


Figure 3.10. Typical solutions drawn in Cartesian space for the drawers scenario from each planner. Circle and triangle markers correspond to the start and end states, respectively.

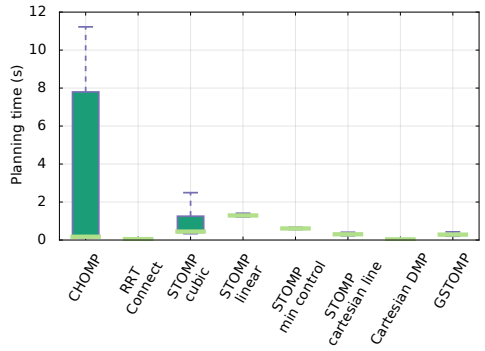
Table 3.5. Drawer-opening scenario: statistics from a total of 600 queries with each planner

Algorithm	Planning time (s)		Smoothness		Success rate
	Median	Std dev	Median	Std dev	
CHOMP	7.909	3.139	**0.000**	0.000	55%
RRTConnect	0.031	0.012	0.029	0.013	**98%**
STOMP cubic	0.608	0.925	0.039	0.015	63%
STOMP linear	0.753	0.719	0.050	0.049	32%
STOMP min control	0.776	0.608	0.024	0.016	57%
STOMP Cartesian line	0.357	0.852	0.082	0.022	38%
Cartesian DMP	**0.028**	0.012	0.018	0.000	21%
GSTOMP	1.25	3.98	0.092	0.522	72%

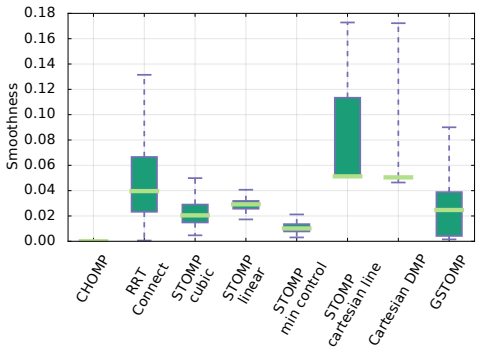
3.2.3 Cupboard opening

While the cupboard opening task also has the *A-to-B* element, the additional constraint here is to follow a curved path. While all planners are able to produce results to the *A-to-B* part of the planning problem, - except for GSTOMP and DMP - they have no knowledge of the curved path constraint. Figure 3.6c depicts the queries used for benchmarking.

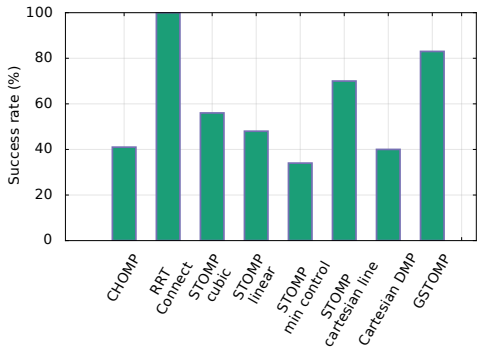
Results of the cupboard opening scenario are presented in Figure 3.11 and Table 3.6.



(a) Planning time, lower values are favoured



(b) Smoothness, lower values are favoured



(c) Success rate, higher values are favoured

Figure 3.11. Cupboard scenario experiment results

One typical trajectory was drawn for each planner in Figure 3.12. The desired trajectories lay in a less confined space than the drawer opening scenario. It can be expected that some planners may produce more elongated paths which is visually assessable in Figure 3.12.

As before, 600 queries were made to each planner. RRTConnect and DMP delivered the fastest performance. GSTOMP and other STOMP variants performed equally good. Again, CHOMP required more planning time and showed a high variance in its performance. GSTOMP performs on par with other methods regarding smoothness, with only CHOMP performing better. RRTConnect performed statistically worse than GSTOMP, while DMP and STOMP with Cartesian initialisation were the worst performers. Regarding the success rate GSTOMP showed a 24% increase in success rate than other STOMP variants while providing trajectories consistent with the demonstration: a slightly curved trajectory with increasing height. As discussed in Section 3.2, this assumes some freedom of the parallel gripper along the handle and is the result

of the provided demonstration. In general the results are consistent with the drawer opening scenario confirming the benefits of combining LfD methods with trajectory optimisation ones.

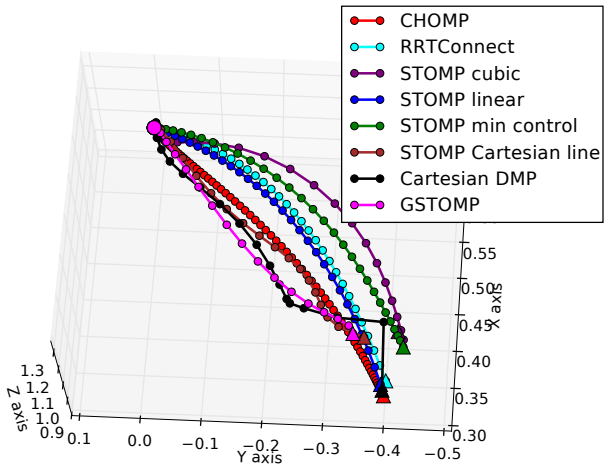


Figure 3.12. Typical solutions for the cupboard scenario drawn in Cartesian space. Circle and triangle markers correspond to the start and end states.

Table 3.6. Cupboard-opening scenario: statistics from a total of 600 queries with each planner

Algorithm	Planning time (s)		Smoothness		Success rate
	Median	Std dev	Median	Std dev	
CHOMP	0.189	4.007	**0.000**	0.000	41%
RRTConnect	0.053	0.042	0.049	0.038	**100%**
STOMP cubic	0.497	1.343	0.023	0.015	56%
STOMP linear	1.301	0.476	0.030	0.006	48%
STOMP min control	0.615	0.321	0.011	0.010	34%
STOMP Cartesian line	0.310	0.137	0.097	0.075	70%
Cartesian DMP	**0.036**	0.011	0.050	0.036	40%
GSTOMP	0.296	0.171	0.036	0.020	83%

3.2.4 Changing the number of DoF

This set of experiments study how planners perform when the number of DoF decline in the cupboard opening scenario. This aims to show the benefits of planning using GSTOMP in multiple manipulators having a different number of degrees of freedom.

In addition this test provides information regarding the skill-transfer between robots, especially if one considers that a robot with artificially reduced DoF is essentially a different platform.

Using the first start/end state pair from the drawer-opening experiment which all planners were most successful on, 200 queries were made with the same set of planners and three different DoF configurations of the same robot:

- the original, 8 DoF *TIAGo* robot consisting of the elevating torso and a 7 DoF arm,
- a fixed torso version, reducing the planning problem to the 7 DoF arm, and
- a 6 DoF version of the arm, fixing the first joint such that it makes start and end states of the task instances possible.

The results presented in Table 3.7, Figure 3.13 and Figure 3.14 follow a similar trend for the 6 DoF case. Surprisingly, RRTConnect didn't succeed at finding a viable solution. This is due to the strongly limited workspace of this arm configuration. The algorithm operates on a collection of start and end nodes to allow for some tolerance and provide good workspace coverage when exploring. Unfortunately, this highly limited workspace challenged sampling start and goal nodes due to failures in inverse kinematics calculations. The rest of the planning algorithms operate mainly in joint-space which in this scenario produced slightly better results except for CHOMP, where even though the optimisation produced kinematically feasible trajectories, all were colliding with the environment. The proposed approach performs on par or better than most of the other approaches. It also shows a smooth degradation to the reduction of DoF. In terms of speed, GSTOMP performs comparably with other STOMP variants. RRTConnect, CHOMP and DMP perform better.

It should be noted how GSTOMP always performed as good as or better than other STOMP variants, clearly showing the benefit of the task-informed *guide* trajectory.

3.2.5 Skill transfer: drawer opening scenario

Transferring skills between robots is a major merit of GSTOMP. In this experiment, the drawer opening skill demonstration was recorded with *TIAGo* but was executed on the *Rob@work3* platform. To provide even ground for comparison, an altered version was created of *Rob@work3*, replacing the *end-effector* with the parallel gripper of *TIAGo* in

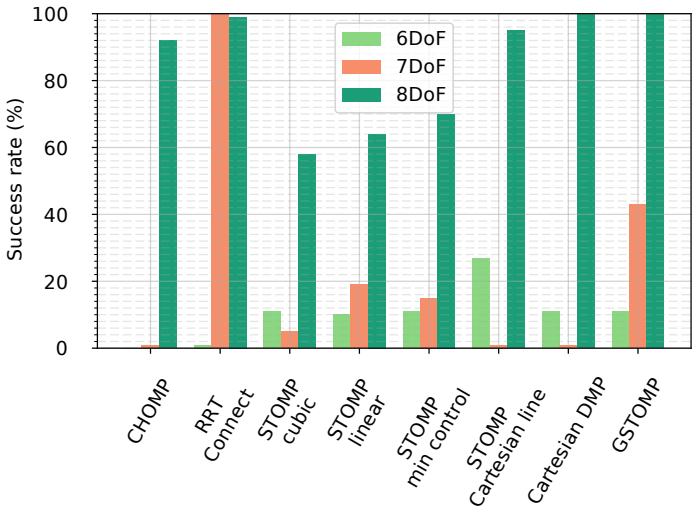


Figure 3.13. Success rate for a single cupboard task with varying number of DoF enabled on the *TIAGo* robot

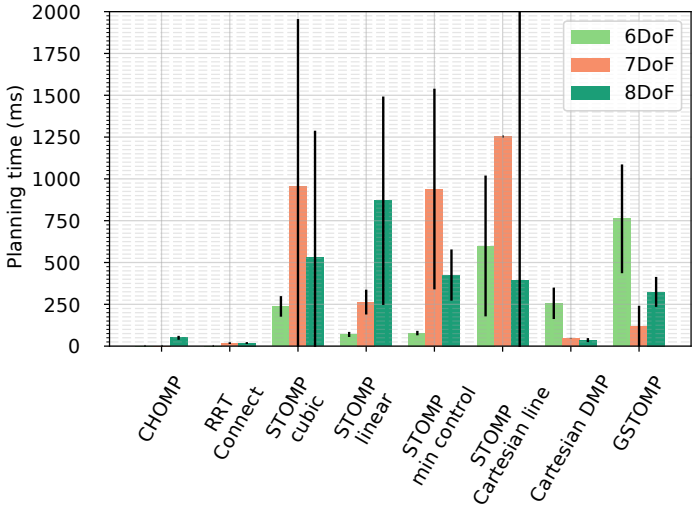


Figure 3.14. Planning time for a single cupboard task with varying number of DoF enabled on the *TIAGo* robot

simulation. Drawer opening was chosen to demonstrate skill transfer due to the limited height of *Rob@work3*. Since generalisation over different drawers was proved before, this experiment only focuses on the top drawer as depicted in Figure 3.15.

To provide the same context, a total of 600 experiments were ran, 100 for each planner. All STOMP variants were set to use 20 samples and were forced to run at least one iteration of the optimisation before terminating to avoid biasing results with "lucky initialisation" for this shorter task. Similarly to CHOMP, STOMP variants where in-

Table 3.7. Varying DoF for cupboard opening: statistics from a total of 600 queries with each planner

Algorithm	Planning time (s)			Success rate		
	6DoF	7DoF	8DoF	6DoF	7DoF	8DoF
CHOMP	N/A	N/A	0.048	0%	0%	92%
RRTConnect	N/A	**0.016**	**0.016**	0%	**100%**	99%
STOMP cubic	0.222	0.785	0.381	11%	5%	58%
STOMP linear	**0.078**	0.238	0.701	10%	19%	64%
STOMP min control	0.083	0.496	0.381	**15%**	**70%**	6%
STOMP Cartesian line	0.599	1.254	0.391	27%	1%	95%
Cartesian DMP	0.256	0.048	0.036	11%	1%	100%
GSTOMP	0.691	0.075	0.311	10%	42%	**100%**

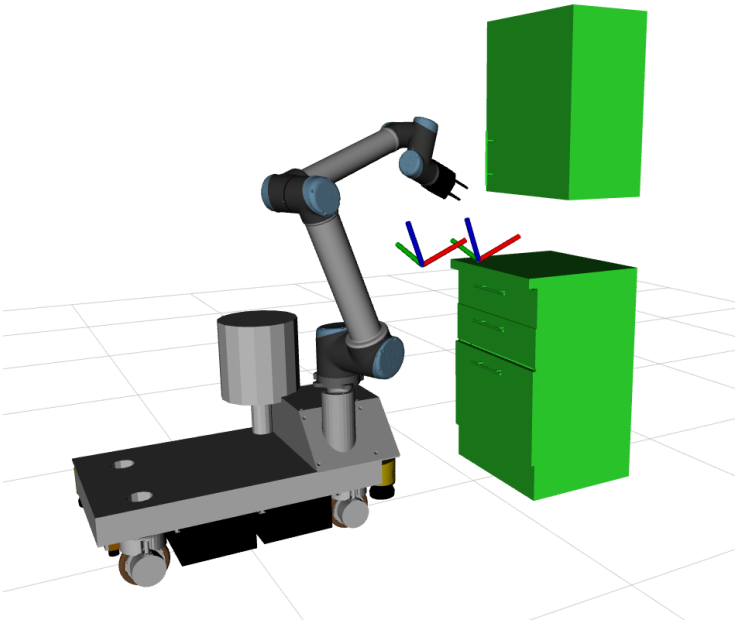


Figure 3.15. Experiment setup for transfer of drawer opening skill

terpolation is inherently smooth, the generated straight paths performed well while initialisation for minimal quadratic control cost generated jagged paths often colliding, resulting in poor performance due to collisions with the environment. GSTOMP performed well in transferring the semi-straight line from the *TIAGo* demonstration. As before, when there is a path of any quality, RRTConnect finds one extremely fast.

In Figure 3.16 it is shown that GSTOMP performs really good on this task, similarly with the performance using the *TIAGo* robot. It is noteworthy that some of the STOMP variants improved drastically using the other platform. Figure 3.17 shows the

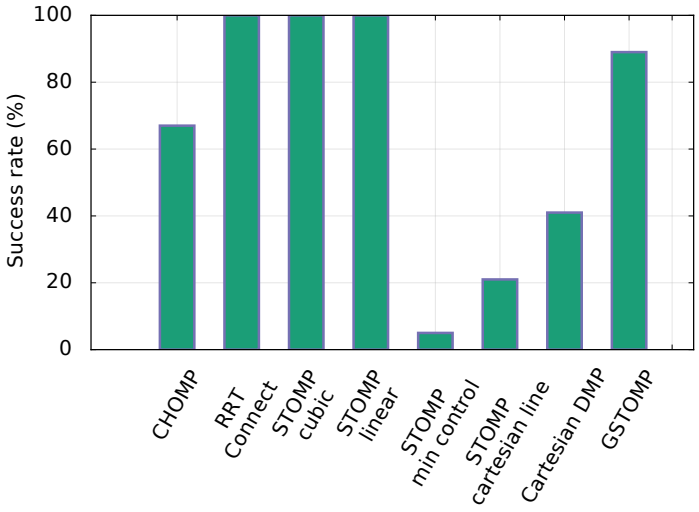


Figure 3.16. Success rate of the skill transfer task

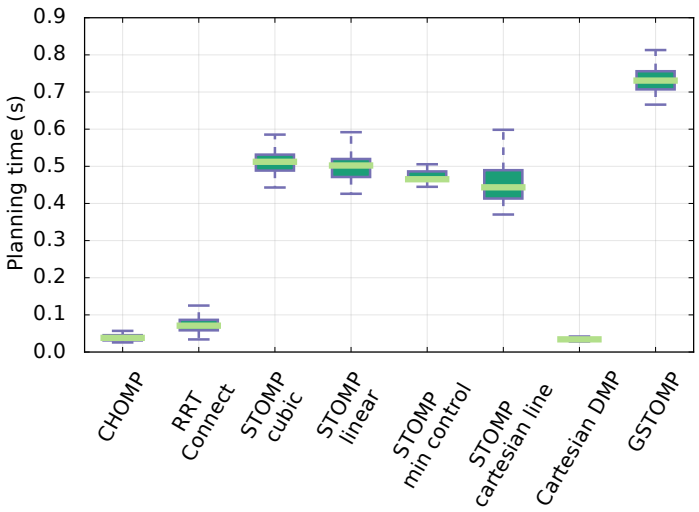


Figure 3.17. Planning time of the skill transfer task

proposed approach performing worse than all the other methods, but still quite close to the STOMP variants. CHOMP, RRTConnect and DMP perfrom much better. Figure 3.18 presents a typical solution of this task by each planner. Circle and triangle markers mark the start and end states. Even though most planners managed to generate paths with relative ease, their shape is typically curved, not suitable for the task. On the other hand, GSTOMP results are relatively straight, following the drawer's movement and achieved this in time comparable to other STOMP variants. Table 3.8 sums up the statistics presented in graphs in a table format.

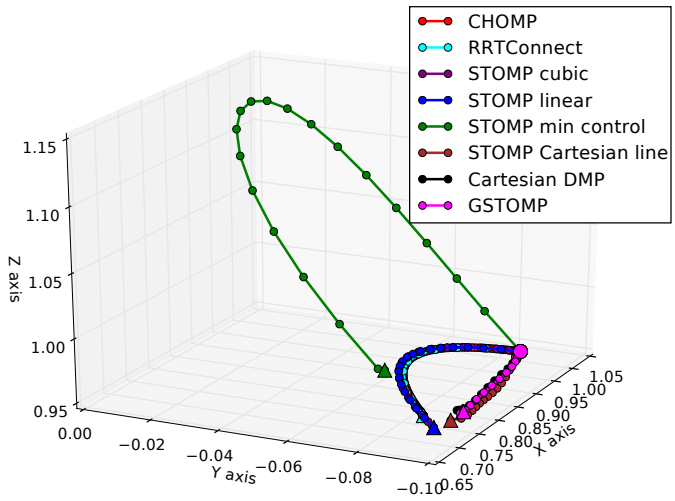


Figure 3.18. Typical trajectories drawn in Cartesian space from each planner in the skill transfer experiments

Table 3.8. Skill transfer scenario: statistics from a total of 600 queries with each planner

Algorithm	Planning time (s)		Success rate
	Median	Std dev	
CHOMP	0.040	0.017	67%
RRTConnect	0.074	0.023	**100%**
STOMP cubic	0.520	0.207	**100%**
STOMP linear	0.508	0.209	**100%**
STOMP min control	0.492	0.022	5%
STOMP Cartesian line	0.467	0.778	21%
Cartesian DMP	**0.035**	0.010	41%
GSTOMP	0.738	0.258	89%

3.3 Discussion

3.3.1 Noise rate

It is possible to fine-tune the standard deviation of the distributions $\mathcal{N}$ that are used for generating noise on each joint in STOMP, as shown in Algorithm 3.2. The default values of these parameters are relatively large, using a standard deviation of 1.0. Using this setting vastly different trajectories were often experienced from *guides*, resulting in slow convergence and high failure rate. It is preferred to take much finer steps when planning in tight, collision-rich spaces. Consequently the default 1.0 standard deviation

was replaced with 0.1 for each joint in the experiments, for all STOMP variants and GSTOMP alike. This ensures a better fitting exploration in the noisy trajectory rollout phase of STOMP and significantly increases the convergence rate of the optimisation process.

3.3.2 The sampling of the *guide trajectory*

The STOMP optimisation works best when points in the trajectory are equidistant. This causes a problem when these trajectories are used directly as *guides* for GSTOMP as the timesteps generated by the Canonical system $\dot{x} = \alpha_x x$ within the DMPs is not equally spaced. As depicted in Figure 3.19a, these trajectories have many points on both ends. The optimisation within STOMP works by adding noise to each individual points of the trajectory. If there are regions within the trajectory that are represented with more points than others, the noise-adding effect of the optimisation will be higher there. With the first implementation it was quickly discovered that 80% of points in a *guide trajectory* were from one of the two ends creating hook-shaped curves in the optimisation process (Figure 3.19b). This can be solved by re-sampling the result of the DMP rollout by using an interpolator and a linearly-spaced set of timesteps

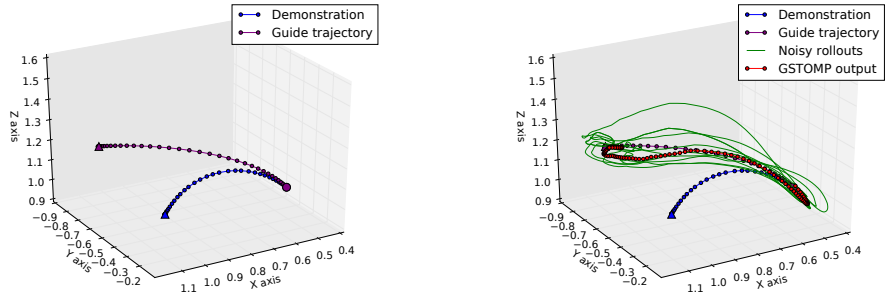
$$[0.00, 0.01, 0.02, ..., 1.0]$$

. This however only guarantees smooth sampling if the velocity profile of filtered trajectories are smooth. This is hardly the case with DMP-s as shown on Figure 3.20 but there exist methods [80] that tackle smoothing and smooth sampling of trajectories. An equi-time resampled guide trajectory will only reduce the density of crowded regions within the trajectory but won't solve the issue.

The guide trajectory should be resampled with fixed euclidean distance and possible fixed Δt.

3.3.3 DMP Canonical system

The Canonical system as presented in [29] Equation 3.25 is similar in shape to a logarithmic function but if implemented via the equation only, the convergence strongly depends on the amount of iterations ran with the equation as well as on the α_x term. In this work's DMP implementation the Canonical System always goes from 1.0 to 0.0 in



(a) The beginning and end of both trajectories are densely sampled

(b) The hooks appear in all noisy rollouts during the optimisation process.

Figure 3.19. The steps of acquiring a solution within GSTOMP

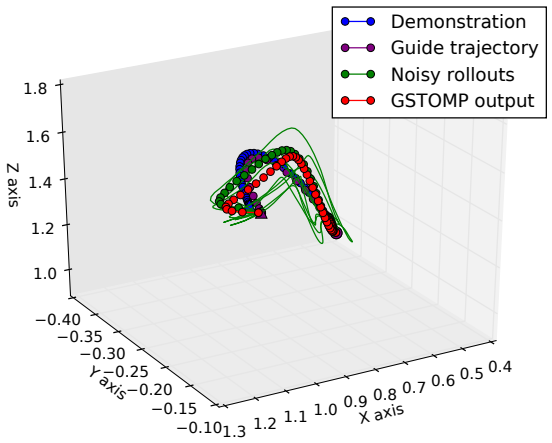


Figure 3.20. An equi-time resampled guide trajectory

order to activate the diminishing term in Equation 3.9.

$$\tau \dot{x} = -\alpha_x x \tag{3.25}$$

The same results can be achieved by sampling from the expression presented in Equation 3.27. Here, $logspace$ generates a list of logarithmically-spaced values from 1.0 to 0.0 with $num\_timesteps$ number of samples. The division by $timesteps[0]$ takes care of numerical instabilities, should the generated vector of $timesteps$ not start at exactly 1.0.

$$timesteps = logspace(1.0, 0.0, num\_timesteps) - 1.0 \tag{3.26}$$

$$timesteps = timesteps/timesteps[0] \tag{3.27}$$

Since the approach of Equation 3.27 does not depend on iterations, the convergence is always guaranteed, regardless of number of samples and start and end states as shown in Figure 3.21

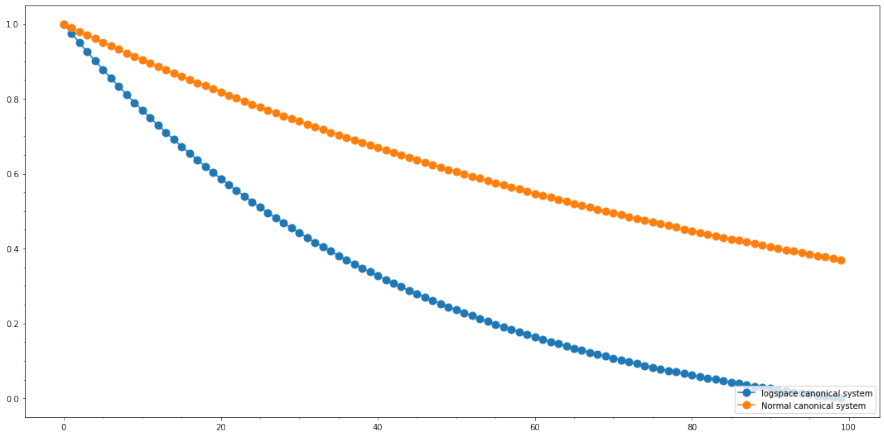


Figure 3.21. Canonical systems comparison

3.4 Summary

This chapter proposed a novel motion planning algorithm GSTOMP, which extends an existing optimisation-based technique STOMP, to use trajectories (*i.e. guides*) for optimisation initialisation. These *guide* trajectories are produced by Cartesian Dynamical Movement Primitives. A new cost function was introduced to the optimiser that penalises moving away from the *guide* trajectory unless it is overruled by other cost functions such as collision and joint limit avoidance. These appear as weighted terms in the optimisation process of GSTOMP.

The proposed system is able to generalise over single demonstrations of tasks for new start and goal states. The *guide trajectories* implicitly encode complexity without the need to design task-specific mathematical models or execution frameworks as in [81]. The demonstrations used to generate *guide trajectories* may be acquired from kinesthetic teaching, motion capture systems or video analysis. GSTOMP does not

cover approach and departure paths as this can be achieved by any planner but rather focuses on generating task-specific trajectories.

It is clear that approaches crafted for specific tasks are likely to perform better, but it was also shown that GSTOMP scales well with the number of tasks. Extensive studies were performed on shelf-picking and drawer- and cupboard opening scenarios and the GSTOMP was compared against a variety of state-of-the-art approaches using the same software framework. It was shown that GSTOMP is able to leverage prior demonstrations of task instances, ensuring correct, task-specific trajectories. This combination improved the success rate over pure STOMP or pure DMP approaches. On the one hand it allowed STOMP to better avoid obstacles and escape minima by using a better initialisation. On the other hand, it also allowed the task specific trajectories to cope with variance in the environment.

GSTOMP was shown to allow for sharing *guide trajectories* between different robots achieving a form of skill-transfer. Since the demonstrations are captured in Cartesian-space, the only challenge for achieving this is kinematic feasibility: robots or humans may produce demonstration trajectories impossible to be tracked by a robot with fewer degrees of freedom or a different kinematic setup. To address this question specifically, a study was conducted on the effects of limited DoF on each planner which confirmed that GSTOMP was always as good as or better than other STOMP variants.

The parameters for joint-space noise distributions used in the *noisy trajectory generation* step of STOMP are currently determined empirically. When learning a task for a given robot, a set of trajectories could be used to learn these parameters, connecting a distribution-based learning approach with the distribution-based optimisation approach.

A possible extension to improve convergence rate of the stochastic optimisation process is to add a maximum budget of total noise. The experimental results of this chapter showed that, at times, stochastic optimisation may fail to converge due to the breadth of exploration in a single iteration. A fixed noise budget would allow the optimisation to iterate more, taking more gradual steps of improvement on the trajectory. The reason for not simply shrinking the distribution of noise applied to all joints lies with the individual, per-joint noise specification. Sampling noise could be done in a random order from each joint, until the budget is depleted, or noise for every joint was sampled. This in effect minimises the block distance, similar to L_1 norm of the initial and noisy trajectory, dynamically reducing the step size of the optimisation.

The current approach uses sampling-based continuous collision checking. In the

work of [54] the concept of *swept out volumes* is introduced. An interesting future direction would be to compare the two collision checking methods in this framework.

Additionally, in the current implementation the robot's mobile base is assumed to be static in front of the task area. Inspiration could be taken from the works of Vahrenkamp *et al.* [27] and Zacharias *et al.* [26] to add mobile base placement to the motion planning and initialisation problem, further improving its success rate, computation time and smoothness.

In summary, this chapter has validated the idea of introducing prior demonstrations of tasks into the motion planning process and proposed a solution to achieve it. Furthermore, an inherent feature of the system, skill-transfer has been shown along with the ability to handle limitations in the manipulator kinematics in a flexible manner. The results of this chapter are in the process of publication as [16]. Thanks to the modular approach, both the task-informed trajectory generator and the optimisation-based planner components can be replaced with something providing the same interfaces and operating with similar assumptions. The following chapters of the thesis will design an alternative optimisation-based planner that improve both the quality of solutions and robustness of the planning algorithm.

Chapter 4

Smooth Mobile Base Trajectory Planning by Optimisation

OPTIMISATION-BASED motion planners often challenge their users with the amount of parameters and their sensitivity towards them. Stochastic optimisation is flexible and can find solutions for a large variety of problems. Unfortunately, the performance of *Stochastic Trajectory Optimization for Motion Planning* (STOMP) was very hard to improve, let alone remain consistent for the *Ambient Assisted Living* (AAL) manipulation motion planning problems it was tested for in Chapter 3. If the distributions are too large, STOMP was found to have issues with convergence and quality of results.

Supporting a large variety of tasks with a method that may require tuning parameters between tasks is a near impossible challenge. Adding new constraints to such system likely extends the parameter space, incurring additional parameter-adjustment overhead. In order to more robustly support scaling the number of tasks in a deployed system as well as new type of constraints, a gradient-based optimisation approach is considered hereafter in the thesis.

This chapter studies a simpler problem to build up the theoretical concepts of a novel, numerical gradient-based motion planner. The motivation for a temporary shift in domain is to explore the possibilities for improving trajectory quality using a gradient-based optimisation motion planner.

The presented method, coined *Timed-Elastic Smooth Curve* (TESC) planner, tackles the problem of mobile base trajectory planning for navigation. The focus of this approach is to provide an optimisation-based solution that requires the least amount of parameter tuning between different scenarios while being able to handle a large set of well-defined quality constraints. To this end, the use of a novel piecewise C^n smooth

curve for mobile-base motion planning and control is proposed. Based on a *Timed Elastic Band*, the problem is defined such that trajectories lie on a spline in $\mathrm{SE}(2)$ with nonvanishing *n-th* derivatives at each point. Formulated as a multi-objective nonlinear optimisation problem, it allows imposing soft constraints such as collision-avoidance, limits (*e.g.* velocity, acceleration and jerk) and more. The planning process is online, allowing the robot to navigate in complex dynamic scenarios.

TESC is compared against the state-of-the-art in different scenarios using the *TIAGo* robot. A decomposition of *TIAGo*'s motion traversing through a scene is depicted in Figure 4.1. In this figure, obstacles are marked by dark regions on the floor and the robot is drawn at a set of time-equidistant points on the trajectory.

Planning safe, predictable trajectories is of fundamental importance for autonomous mobile robots to enable high-level applications in service robotics but also for autonomous transportation. Not only should a robot be able to move towards a desired goal, but it should do it optimally, and with a level of safety. Additionally, if the use-case and environment have dynamic aspects, planning must happen online in order to react to changes.

Splines, specifically with nonvanishing *n-th* derivatives facilitate defining constraints such as velocity, acceleration and also jerk limits, which is a desirable aspect of predictable robot behaviour but have other benefits, for example, to improve comfort and prevent motion sickness [82] in autonomous people transportation vehicles.

Differential geometry for trajectory optimisation has often been used in robotics. Twenty years ago, Zefran et al. [83] proposed smooth rigid-body motion on $\mathrm{SE}(3)$ using Lie groups. The most common approaches for estimating piecewise-smooth curves are based on - in order of increasing complexity - piecewise polynomial functions [84]–[86], B-splines [87], [88] or *Non-Uniform Rational B-splines* (NURBS) [89], [90]. Alternative models leverage Bezier Curves geometrically constructed on Manifolds [91], unfortunately this approach is not continuous at spline knots, therefore derivatives are not guaranteed at these points.

This chapter describes the *Timed-Elastic Smooth Curve* (TESC) planner for mobile-base motion planning and control. Building upon the work of Rösmann *et al.* [15], [63] a sparse trajectory planning scheme is formulated, whose discretisation lies on piecewise C^n curve on a Lie manifold. Unlike the work of [15], [63] where the connectedness of *Timed Elastic Bands* (TEB) is maintained by the use of kinematic constraints, this novel approach achieves it by a revisited formulation of smooth interpolation on Lie

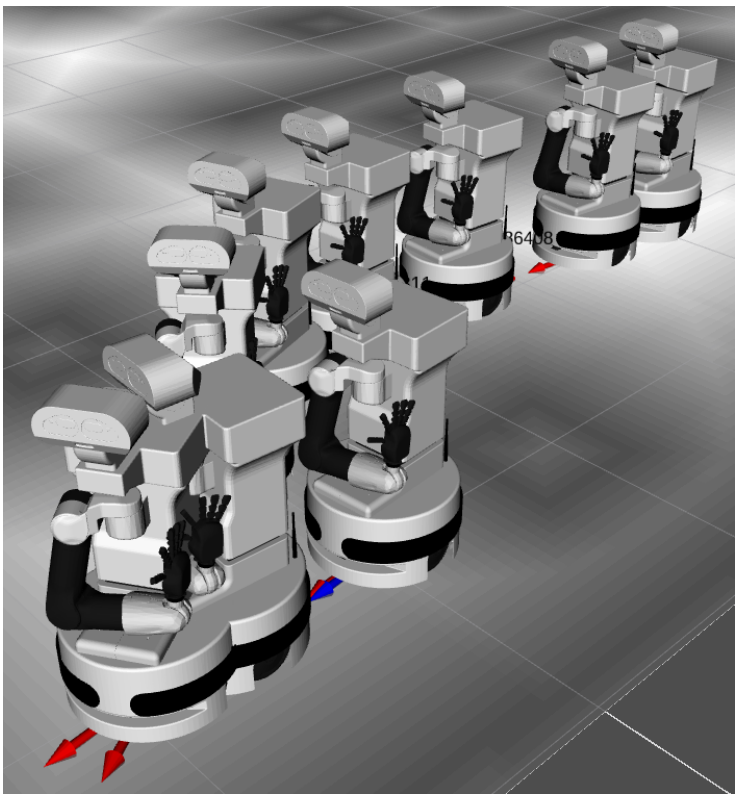


Figure 4.1. Decomposition of TIAGo's path

manifolds. This ensures nonvanishing *n-th* derivatives at any point along the trajectory. Furthermore, the TESC approach proposes a new take on trajectory estimation, as opposed to the traditional polynomial or spline-based trajectory estimation, this algorithm does not aim to estimate coefficients [87] nor control points which define the curve[91], [92]. Instead, TESC uses a discrete collection of points which themselves lie on the piecewise curve in $\mathrm{SE}(2)$ that forms the trajectory. The formulation of the problem allows for a seamless calculus of a pose at any time along the TEB. Such a continuous trajectory representation ensures that constraints are not subject to error from discretisation. Additionally, the resulting trajectory may be resampled with ease to an arbitrary resolution which otherwise would only be possible by increasing the number of points in the optimisation, therefore increasing the computational complexity.

The rest of this chapter is organised as follows. The TESC planner formulation is detailed in Section 4.1. Section 4.2 describes the experiments and results. Section 4.3 draws a conclusion and proposes further developments.

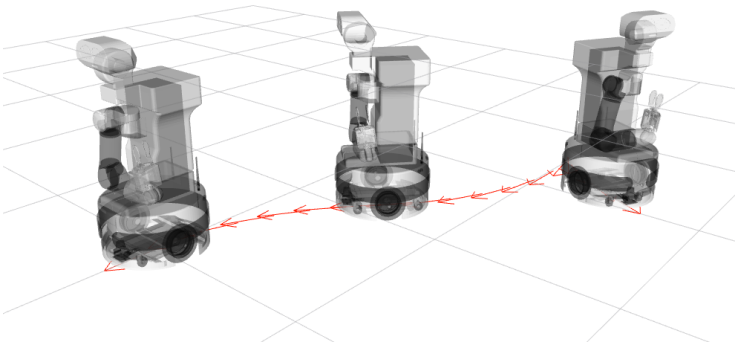


Figure 4.2. *TIAGo*'s path from the side

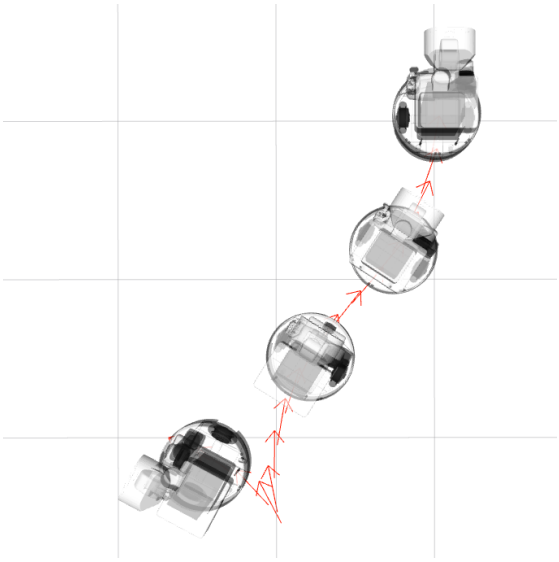


Figure 4.3. *TIAGo*'s path from above

4.1 TESC Planning

In this section, an alternative representation of task-space trajectory is used to facilitate the definition of specific quality constraints. Extending the concepts introduced in Section 2.2, this alternative representation defines the trajectory as a sequence of points lying on a Lie manifold $\mathrm{SE}(n)$ of rigid motions. Although this formulation can be expressed in a general form, defined for an arbitrary number of dimensions n, this section focuses on the algebraic realisation for the Lie manifold $\mathrm{SE}(2)$ in the plane, that is, translation in the plane, rotation over the z-axis, which is where all benchmarking experiments are performed. The tools are beyond the scope of this thesis and a didactic review of Lie Algebra can be found in [93]. However, a comprehensive summary is

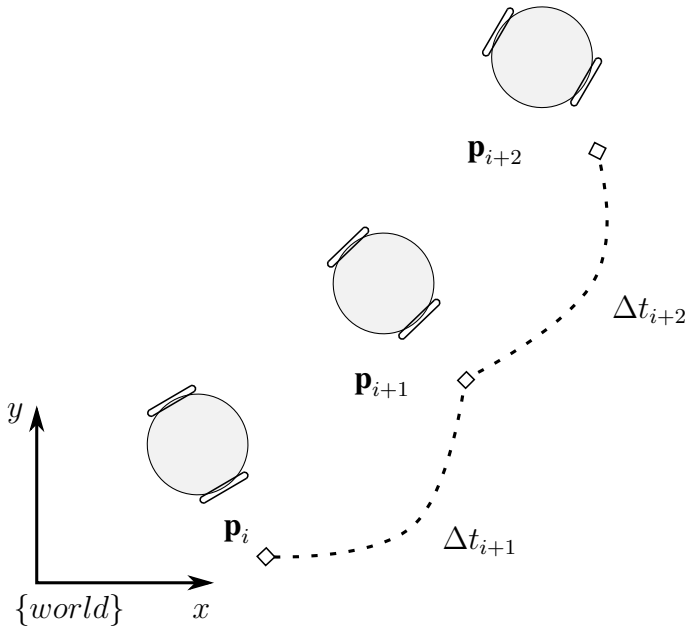


Figure 4.4. Robot path described by a set of robot poses

included in Section A.1.

4.1.1 Problem Formulation

Timed-Elastic Band

The Timed-Elastic-Band is described as a sequence of $n + 1$ robot poses $\in$ SE(2) forming a chain between an initial and a final configuration such that

$$\mathbf{Q} = \{\mathbf{p}_i\}_{i=0\ldots n}. \tag{4.1}$$

For the case of mobile-base planning, the initial configuration, noted as $\mathbf{p}_0$, is fixed at the origin. All consecutive poses are tied to one another by n time intervals Δt,

$$\Delta\mathbf{T} = \{\Delta t_i\}_{i=1\ldots n}\ , \tag{4.2}$$

where Δt_i denotes the time interval required for the robot to move from a pose $\mathbf{p}_{i-1}$ to $\mathbf{p}_i$ along the trajectory. Such a Timed-Elastic Band constructed this way is illustrated in Figure 4.4. The elastic property is provided by a collection of soft constraints defined on neighbourhoods poses.

Multi-objective Problem

Defining the pairs,

$$\mathbf{P} = \{(\mathbf{p}_i, \Delta t_i)\}_{i=1...n} \ , \tag{4.3}$$

the TEB is formulated as a multi-objective optimisation problem:

$$\mathbf{P}^* = \underset{\mathbf{P}}{\operatorname{argmin}} \sum_{k,i} w_k c_{ki}(\mathbf{Q}_i, \Delta \mathbf{T}_i) \ , \tag{4.4}$$

which is the type of formulation that can be solved by means of a least-squares nonlinear solver. The choice of the optimisation solver was motivated by the requirements of onboard deployment and smaller computational complexity. The different cost contributions $c_{ki}(\mathbf{Q}_i, \Delta \mathbf{T}_i)$ are balanced by weight factor terms w_k. These costs are computed from $\mathbf{Q}_i$ and $\Delta \mathbf{T}_i$, subsets of $\mathbf{Q}$ and $\Delta \mathbf{T}$ respectively, in the neighborhood of Δt_i. The subsets are required to capture the inputs of cost functions which do not always require elements from $\mathbf{P}$ but instead directly from $\mathbf{Q}$ and $\mathbf{T}$. For instance, deriving velocity requires $\mathbf{p}_i, \mathbf{p}_{i+1}$ and Δt_i. The cost functions are built simply with $c_k = \mathbf{e}_k^\top \mathbf{e}_k$, with $\mathbf{e}_k$ an error measure. In this chapter, error measures use a set of indices $k \in \{s, v, a, j, l, g, o\}$ indicating the type of each objective error during the interval Δt_i. The subscripts are derived from the corresponding error names, namely, **s**moothness, **v**elocity limit, **a**cceleration limit, **j**erk limit, **l**ength and time, **g**oal tolerance, and **o**bstacle avoidance. Following, the different types of objectives are described with the time index i omitted in the notations for brevity.

$\mathbf{C}^n$ Smooth Curve

This objective function considers the trajectory as a collection of discrete points, and aims at enforcing consecutive points to lie on a smooth curve in $\mathrm{SE}(2)$, and that different pieces form a continous spline. As proposed by Jakubiak *et al.* [94], a geometric two-step algorithm can generate smooth splines on Riemannian manifolds, in particular Lie groups. For any pair of consecutive robot configurations $\mathbf{p}_i$ and $\mathbf{p}_{i+1}$, the algorithm interpolates an intermediate configuration $\mathbf{p}_t$ (with $t \in [i, i+1]$) such that it lies on a smooth curve connecting $\mathbf{p}_i$ and $\mathbf{p}_{i+1}$. Leveraging this interpolation algorithm, the smoothness objective function uses three consecutive poses $\mathbf{p}_{i-1}$, $\mathbf{p}_i$, $\mathbf{p}_{i+1}$, and their associated time intervals Δt_i and Δt_{i+1} as follows. First, tangent vectors $\mathbf{v_{i-1}}$ and $\mathbf{v_{i+1}}$

are computed by approximation with backward differences and the interpolation factor t,

$$\mathbf{v_i} = \mathbf{p}_i \ominus \mathbf{p}_{i-1} \tag{4.5}$$

$$t = \frac{\Delta t_i}{(\Delta t_i + \Delta t_{i+1})} \in [0, 1] \tag{4.6}$$

Then, the desired interpolated point $\hat{\mathbf{p}}_i$ is computed as

$$\mathbf{l}(t) = \mathbf{p}_{i-1} \oplus (t \cdot \mathbf{v_{i-1}}) \,, \tag{4.7}$$

$$\mathbf{r}(t) = \mathbf{p}_{i+1} \oplus ((t-1) \cdot \mathbf{v_{i+1}}) \,, \tag{4.8}$$

$$\boldsymbol{\rho}(t) = \mathbf{r}(t) \ominus \mathbf{l}(t) \,, \tag{4.9}$$

$$\hat{\mathbf{p}}_i = \mathbf{l}(t) \oplus (\phi(t) \cdot \boldsymbol{\rho}(t)) \,. \tag{4.10}$$

The resulting error can be defined as

$$\mathbf{e}_s = \hat{\mathbf{p}}_i \ominus \mathbf{p}_i \,. \tag{4.11}$$

The smooth aspect of Equation 4.10 lies partly in the real valued smoothing function $\phi(t)$,

$$\phi(s) = \gamma \sum_{j=0}^{m} \frac{a_{m+1+j}}{m+1+j} s^{m+1+j} \,, \tag{4.12}$$

with

$$a_{m+1+j} = (-1)^j \binom{m}{j} s^{m+j} \,, \tag{4.13}$$

$$\gamma^{-1} = \sum_{j=0}^{m} \frac{a_{m+1+j}}{m+1+j} \,, \tag{4.14}$$

where m is the smoothness degree (C^m). It satisfies the following required conditions to guarantee C^m smoothness,

- $[0, 1] \rightarrow [0, 1]$ and strictly increasing,
- zero-valued derivatives at both $\mathbf{t = 0}$ and $\mathbf{t = 1}$.

A more detailed reference of its critical guarantees can be found in [94]. Figure 4.5 illustrates the effect of Equations 4.7–4.11 and how the intermediate (middle) pose $\mathbf{p}_i$ is

attracted onto the smooth curve defined by $\mathbf{p}_{i-1}, \mathbf{p}_{i+1}, \mathbf{v_{i-1}}, \mathbf{v_{i+1}}$ and s. Poses and their corresponding tangents are illustrated by rectangles and arrows respectively.

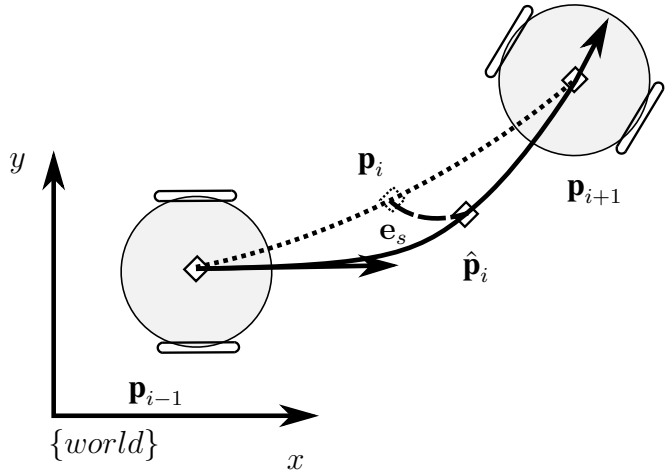


Figure 4.5. Smooth curve constraint

Since the final tangent of the i-th segment associated to Δt_i is to be equal to the initial tangent of the $i+1$-th consecutive segment, associated to Δt_{i+1}, (*i.e.* $\mathbf{v_{i+1,\Delta t_i}} = \mathbf{v_{i,\Delta t_{i+1}}}$), trajectory smoothness at segment junctions (knots) is implicitly ensured. The formulation therefore does not necessitate an explicit equality constraint on the knots, unlike how it is common practice with other spline-based frameworks[95].

Boundary Constraints of Curve Derivatives

The formulation in Section 4.1.1 motivates nonvanishing *n-th* derivatives at every point, which in turn facilitates enforcing limits (*i.e.* inequality constraints) on the derivatives of the curve. In this chapter, such derivatives subject to inequality constraints are:

- $\mathbf{v}_k$ the mean velocity over a Δt
- $\mathbf{a}_k$ the mean acceleration over a Δt
- $\mathbf{j}_k$ the mean jerk over a Δt .

These derivatives are computed using backward finite differencing with a sliding window over a neighbourhood of the past two, three and four configurations, respectively,

$$\mathbf{v}_i = \frac{\mathbf{p}_i \ominus \mathbf{p}_{i-1}}{\Delta t_i} , \tag{4.15}$$

$$\mathbf{a}_i = 2 \cdot \frac{\mathbf{v}_i - \mathbf{v}_{i-1}}{\Delta t_i + \Delta t_{i-1}} , \tag{4.16}$$

$$\mathbf{j}_i = 6 \cdot \frac{\mathbf{a}_i - \mathbf{a}_{i-1}}{\Delta t_i + \Delta t_{i-1} + \Delta t_{i-2}} \, . \tag{4.17}$$

Similarly to [63] the proposed method uses a nonlinear least-squares solver to optimise the TESC problem. The inequality constraints are approximated by two-sided quadratic penalties such that,

$$e_\nu = \beta(\nu, \nu^{\mathrm{L}}, \nu^{\mathrm{U}}) = \begin{cases} -\nu + \nu^{\mathrm{L}}, & \text{if } \nu < \nu^{\mathrm{L}} \\ +\nu - \nu^{\mathrm{U}}, & \text{if } \nu > \nu^{\mathrm{U}} \\ 0, & \text{otherwise} \end{cases} \tag{4.18}$$

with ν a constrained variable, ν^{L} and ν^{U} respectively ν's lower and upper bounds, and $<, >$ are element-wise comparisons. The least-squares optimisation framework used in this chapter supports equality constraints by setting both the lower and the upper bounds to the same value. From Equations 4.15-4.18 results the following error functions:

$$\mathbf{e}_v \text{ subject to } \mathbf{v}^L < \mathbf{v}_i < \mathbf{v}^U \, , \tag{4.19}$$

$$\mathbf{e}_a \text{ subject to } \mathbf{a}^L < \mathbf{a}_i < \mathbf{a}^U \, , \tag{4.20}$$

$$\mathbf{e}_j \text{ subject to } \mathbf{j}^L < \mathbf{j}_i < \mathbf{j}^U \, . \tag{4.21}$$

The cost functions are built simply with $c_\nu = \mathbf{e}_\nu^\top \mathbf{e}_\nu$ for $\nu \in \{v, a, j\}$.

Nonholonomic Constraints

Since the target application of this planner is a mobile-base, the appropriate kinematics constraints need to be taken into account. For instance, mobile-bases with a differential-drive or a bicycle-model are unable to move sideways. In mobile robot literature this is often referred to as the nonholonomic constraint. For this chapter it is expressed as,

$$e_h \text{ subject to } \mathbf{v}_{yi} = 0 \, . \tag{4.22}$$

with $\mathbf{v}_{yi}$ the y-component of the velocity vector computed from 4.15.

Minimum Turning Radius Constraints

For bicycle-model (*e.g.* car-like) mobile-bases, the turning radius is limited by the steering limits. However, the equivalent condition on the inverse turning radius is im-

plemented, since unlike R, $1/R$ changes sign continuously as the robot changes heading from left to right. Defining,

$$\frac{1}{R} = \frac{\mathbf{v}_{\omega i}}{\mathbf{v}_{xi}} \,, \tag{4.23}$$

$$e_r \;\text{ subject to }\; -\frac{1}{R_{min}} < \frac{1}{R} < \frac{1}{R_{min}} \,, \tag{4.24}$$

with $\mathbf{v}_{xi}$ and $\mathbf{v}_{\omega i}$ respectively the x- and the angular-components of the velocity vector 4.15. Mobile-bases that adhere to the bicycle model cannot turn in place as that would entail a turning radius of 0. To handle the special case of $\mathbf{v}_{xi} \to 0$, $\mathbf{v}_{\omega i}$ is constrained to 0 in a similar fashion as Equation 4.22.

Trajectory Execution Time and Length

Execution time and travelled distance are both important quality metrics of a mobile-base trajectory. A weighted-sum of distance and time is intuitive,

$$c = w_l \left\langle \mathbf{v}, \mathbf{v} \right\rangle + \mathbf{w_t}\, \mathbf{\Delta t^2} \tag{4.25}$$

however, the experiments of this chapter revealed that the optimisation framework produces better gradients - therefore performs better - with a different combined approach.

Let the joint length-time error be defined as,

$$\mathbf{v_i} = \mathbf{p}_i \ominus \mathbf{p}_{i-1} \,, \tag{4.26}$$

$$e_l = \left\langle \mathbf{v_i \Delta t}, \mathbf{v_i \Delta t} \right\rangle = \left\langle \mathbf{v_i}, \mathbf{v_i} \right\rangle \mathbf{\Delta t^2} \,, \tag{4.27}$$

with $\langle \,, \rangle$ denoting the inner product as defined in Equation A.8 and $\langle \mathbf{v_i}, \mathbf{v_i} \rangle$ is the arc-length of the i-th segment squared. Using this type of combination, the two objectives enforce each other, giving better results in consistency of solutions. Although this cost function has no particular physical meaning, its benefit lies in being better suited for a numerical optimisation process. Using two different cost functions for time and length, with independent weights as in Equation 4.25, leads to optimising $2(n-1)$ unique cost functions, each contributing with a fairly large error residual with respect to other cost functions. For instance, the metric on time will not exhibit as large of a change as the

distance metric. Instead, merging the two cost functions into 4.26, the optimisation process only has to deal with $n-1$ unique cost functions, leaving a single weight to tune and a residual of a similar order as the others.

Distance to the Target Configuration

While previous cost functions constrain the overall shape of the trajectory, this one motivates the optimisation to align $\mathbf{p}_n$ (the end of the trajectory) with the desired configuration or *goal*, $\mathbf{p}_g$. This objective only applies to the last node of the trajectory as such:

$$\mathbf{e}_g = \mathbf{p}_g \ominus \mathbf{p}_n \,. \tag{4.28}$$

Obstacle Avoidance

In mobile-base navigation scenarios, the planning environment is often represented by a 2D *Occupancy Grid* (OG). In its simplest form, the OG may take distinct values corresponding to whether a cell is free (no obstacles), occupied (obstacle) or unknown. In the scope of this chapter, the environment is represented by an *Euclidean Distance Grid* (EDG) which caches the distance to the closest obstacle in each cell of the grid with obstacle cells having a zero value. Although this approach assumes that the environment can be accurately represented with a grid (which is the case for mobile-bases) it allows to efficiently evaluate whether a configuration is in collision with an obstacle or not. This representation allows for computing gradients so that it fits in TESC optimization scheme. As described by Felzenszwalb *et al.* [96], a 2D-OG, specifically. an EDG can be computed using the distance transform algorithm. Their proposed solution is ideal for this chapter as it is fast, efficient and computes an exact Euclidean distance. The evaluation process starts with a distance grid and two consecutive poses $\mathbf{p}_i$ and $\mathbf{p}_{i+1}$ along the trajectory. Then, k poses are interpolated between $\mathbf{p}_i$ and $\mathbf{p}_{i+1}$ as per [94], with k chosen in adequacy with the grid resolution. This intermediate sampling strategy implements continuous collision-checking from otherwise discreet trajectories and is often a parameter of motion planners using such a method. The EDG cell corresponding to each of the $k+2$ poses are evaluated. Finally, the error is calculated

by

$$e_o = \begin{cases} r - d, & \text{if } d \leq r \\ 0, & \text{otherwise} \end{cases}, \tag{4.29}$$

where d is the minimum cell value, the distance of the closest obstacle, over the $k + 2$ poses and r is the radius of the robot footprint circle. For simplicity, this work does not involve handling complex footprint shapes. By evaluating the intermediate interpolated poses, a continuous collision checking method is yielded. This ensures that each segment is obstacle-free, avoiding the common problem of landing consecutive poses on the boundary of an obstacle and failing to notice an intermediate collision. For the scope of this work, the number of intermediate poses checked for collision can be derived from the grid resolution and the footprint radius r.

4.2 Experiments

This section describes the experimental setup created for this chapter and results from simulations of three different scenarios using the *TIAGo* mobile-manipulator robot.

The proposed TESC approach is compared against the state-of-the-art TEB planner [63] using a series of metrics aimed at highlighting practical and quality aspects. Since the goal of this chapter is to improve on guarantees provided by optimisation-based planners, *Rapidly-exploring Random Tree* (RRT) approaches are not presented in state-of-the-art comparison.

Each planning problem is executed 1000 times on each planner and results are compiled from these experiment runs. To provide a fair ground for comparison, and taking benefit of the likeness of the two compared approaches, TESC and TEB are configured with the same velocity and acceleration limits. The specific parametrisation used in all experiments is presented in Table 4.1, note that TESC uses the same set of parameters for all scenarios.

The evaluation covers eight metrics. First, *success rate* indicates whether the planner has found a collision-free trajectory. *Optimisation time* captures the amount of time the planner took to find a trajectory. Solving the entire problem in a timely manner is very important, especially for reactive control applications. *Trajectory arc-length* and *trajectory time* show how much the robot has to move to reach the goal and how much

Table 4.1. Parameter values used in all experiments with the *TIAGo* robot

Name	Value	Name	Value
$\mathbf{w}_v$	1	$\mathbf{v}^L$ and $\mathbf{v}^U$	x: (-0.6, 0.8), y: (0.0, 0.0), yaw: (-0.4, 0.4)
$\mathbf{w}_a$	1	$\mathbf{a}^L$ and $\mathbf{a}^U$	x: (-1.0, 1.0), y: (-1.0, 1.0), yaw: (-0.8, 0.8)
$\mathbf{w}_j$	1	$\mathbf{j}^L$ and $\mathbf{j}^U$	x: (-1.0, 1.0), y: (0.0, 0.0), yaw: (-1.0, 1.0)
$\mathbf{w}_h$	10	$\mathbf{w}_r$	10
$\mathbf{w}_g$	20	$\mathbf{w}_c$	1
$\mathbf{w}_o$	1	$\mathbf{w}_l$	1

time it takes to get there. *Average velocity* and *average acceleration* metrics include both their linear and angular components. Finally, energy usage is also approximated using a *kinetic-energy* formulation, while *trajectory curvature* serves as an indicator of smoothness. Smoother trajectories require less acceleration/deceleration during execution, as such they impose less stress on the mechanical parts of the robot. Moreover, smooth maneuvres from a robot are generally more predictable, allowing people to both feel and be safer in the robot's environment during operation. These metrics are summarised in Table 4.2.

The energy metric is defined as,

$$\sum_{i=0\ldots N-2} \frac{||\log(\mathbf{p}_i^{-1} \cdot \mathbf{p}_{i-1})||}{2 \cdot \Delta t_i^2} \tag{4.30}$$

The curvature of a trajectory is approximated as the sum of the norm of acceleration in a global frame,

$$\sum_{i=0\ldots N-3} ||2 \cdot \frac{\log(\mathbf{p}_i^{-1} \cdot \mathbf{p}_{i-1}) - \log(\mathbf{p}_{i-1}^{-1} \cdot \mathbf{p}_{i-2})}{\Delta t_i + \Delta t_{i-1} + \Delta t_{i-2}}|| \tag{4.31}$$

The weight factors w_k used by the TESC planner were empirically determined such that costs $w_k c_{ki}(\mathbf{Q}_i, \Delta\mathbf{T}_i)$ would all lie in the same order of magnitude. The tuning of weights is typically done only a single time, once the order of magnitude of a component was well established. It was found that the weights associated to both kinematics (Section 4.1.1) and the goal (Section 4.1.1) must be an order of magnitude higher than other weights, the same observation was made in [63]. The weight factors used for the TEB planner in these experiments are those presented as optimal in the original paper.

Table 4.2. Metrics used in the experiments of this chapter.

Metric	Description
Success rate in (%)	$100 \cdot \frac{success}{success+failure}$
Planning time (ms)	
Trajectory arc length	$\sum \|\|\mathbf{p}_i \ominus \mathbf{p}_{i-1}\|\|$
Trajectory time (s)	$\sum \Delta t_i$
Average velocity	mean$(\|\|\mathbf{v}_i\|\|)$
Average acceleration	mean$(\|\|\mathbf{a}_i\|\|)$
Energy	See Equation 4.30
Trajectory curvature	See Equation 4.31

4.2.1 Obstacle-free planning

In the first scenario, the robot is located at the center of an obstacle-free grid of size 8×8m and is tasked to plan a trajectory for a random-generated goal on this grid. The experimental setup is illustrated in Figure 4.6 with four different goals.

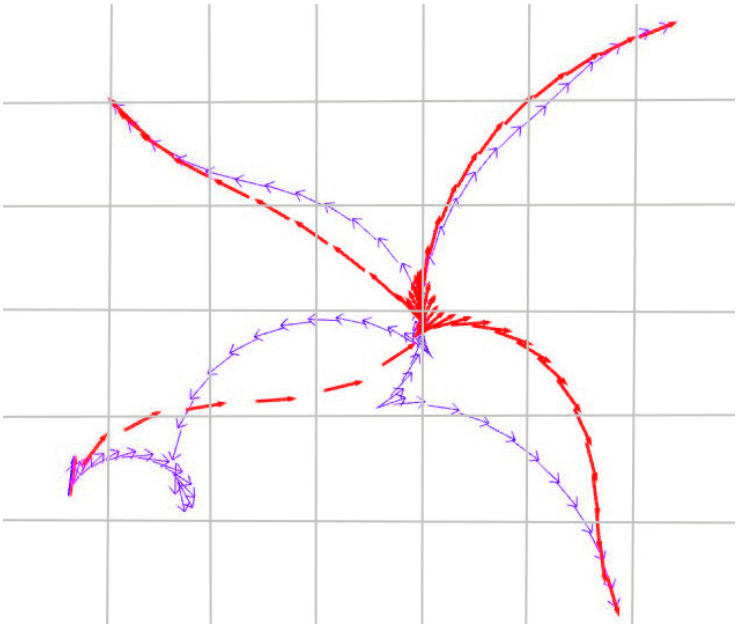
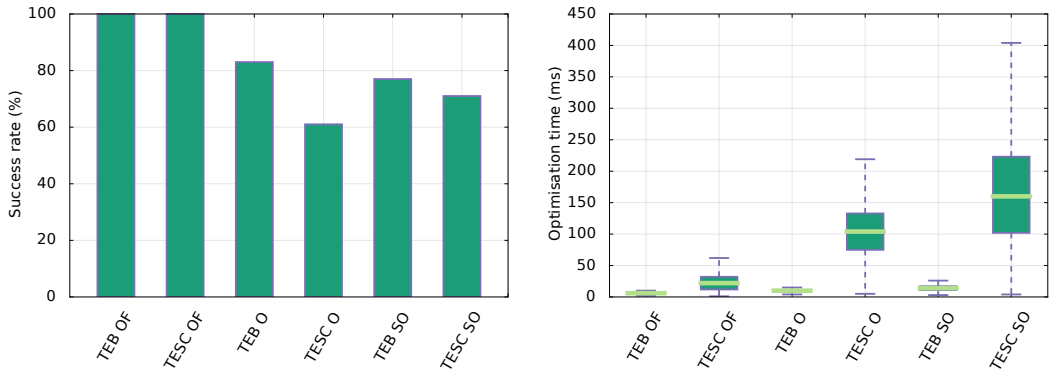


Figure 4.6. Obstacle-free scenario, four different queries. TEB and TESC paths are depicted with violet and red arrows, respectively.

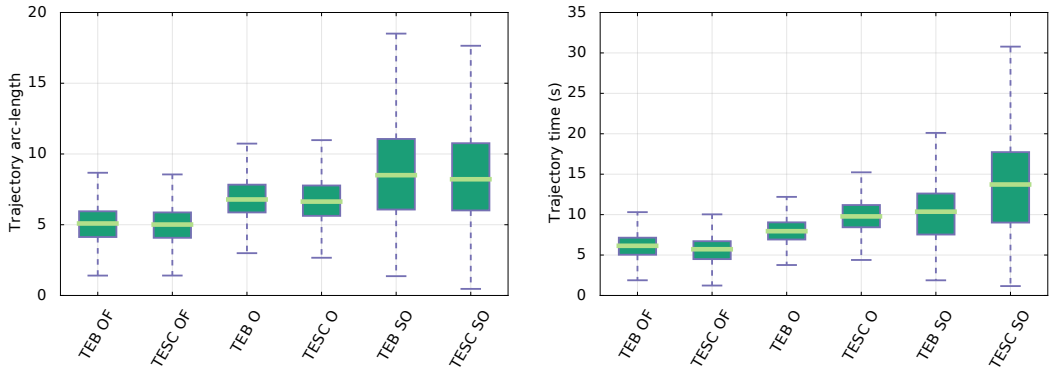
The results of the defined metrics were summarised in Figure 4.7-Figure 4.8, columns associated with this experiment are marked with the **OF** suffix (Obstacle-Free). As Figure 4.7a and Figure 4.7b show, success rate and optimisation time of both planners in this environment is well within the ranges of online capability. Planners performed on par in the trajectory arc length metric, with a small advantage for TESC (Figure 4.7c).

Trajectory time statistics shown in Figure 4.7d are also very similar. For the average velocity metric shown in Figure 4.8a, TESC has a slight advantage compared to TEB but with a much lower acceleration average (Figure 4.8b), which is a more defining factor for motion sickness in the context of autonomous vehicles [82]. Lastly, TESC outperformed TEB in both the total energy cost and the trajectory curvature metrics, shown respectively in Figure 4.8c and in Figure 4.8d. This first experiment highlights that the smooth properties of the TESC approach improve the quality of the generated trajectories while generally matching capabilities of the TEB planner.



(a) Success rate (%), higher values are favoured

(b) Optimisation time (ms), lower values are favoured



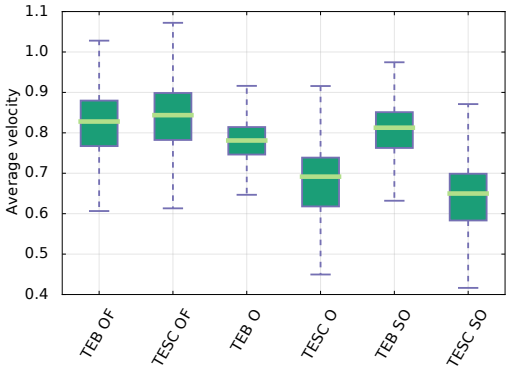
(c) Trajectory arc length (m), lower values are favoured

(d) Trajectory time (s), lower values are favoured

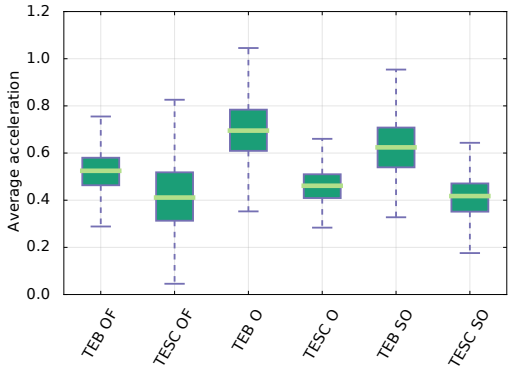
Figure 4.7. Experiment metrics

4.2.2 Synthetic obstacles scenario

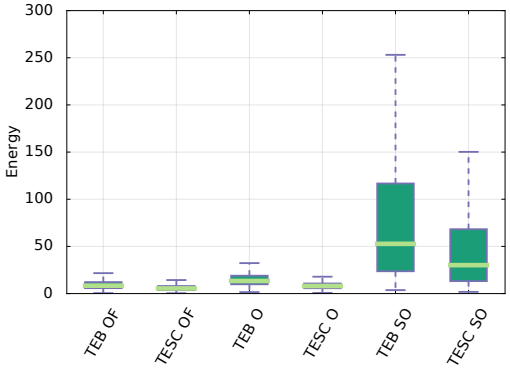
The second environment consists of the same 8×8 meters grid with four round obstacles placed around the robot. Goals are randomly generated and the results for each planner



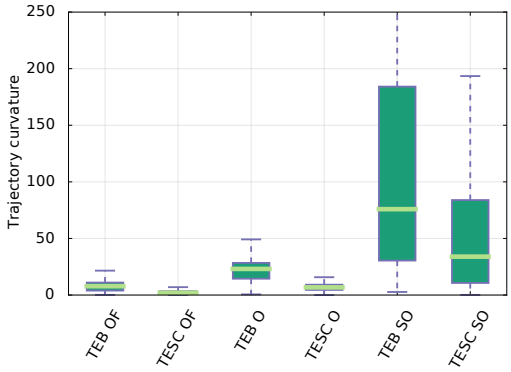
(a) Mean velocity ($\frac{m}{s}$), lower values are favoured



(b) Mean acceleration ($\frac{m}{s^2}$), lower values are favoured



(c) Energy, lower values are favoured

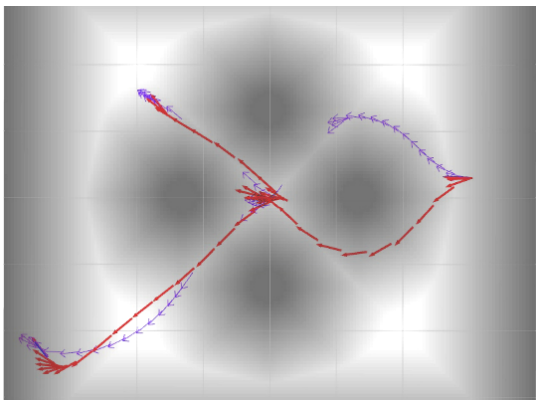


(d) Trajectory curvature, lower values are favoured

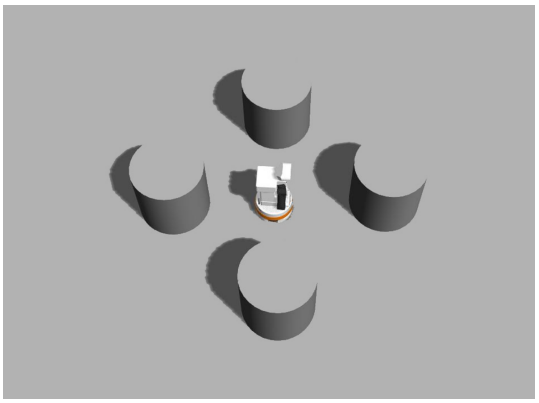
Figure 4.8. Experiment metrics

are feasibility-checked. In Figure 4.9, this scenario is drawn with three different goals. Statistics of the defined metrics were summarised in Figure 4.7-Figure 4.8, columns associated with this experiment are marked with **O** suffix (Obstacles).

The increased problem complexity is reflected by both a degradation of success rates as well as an higher execution time. As shown in Figure 4.7a, both TEB and TESC planner struggled with the problems, successfully producing results for only 83% and 61% of queries respectively. An example failure case of TEB is illustrated on the right side of Figure 4.9, depicting a discontinued trajectory. The continuous collision formulation of TESC defined in Section 4.1.1 aims to avoid exactly this phenomenon. The time taken for planning on this problem increased for TESC while TEB's remains fairly stable as shown in Figure 4.7b. Both planners performed in a similar manner in terms of trajectory time. However, due to the combined objective function, TESC bested TEB in the metrics visible in Figure 4.7c and Figure 4.7d. With a lower average



(a) Occupancy grid view with three different queries.



(b) The four obstacles scenario in Gazebo

Figure 4.9. Four obstacles scenario, TEB and TESC paths are depicted with violet and red arrows, respectively.

velocity (Figure 4.8a) and a much smaller acceleration (Figure 4.8b) TESC produced trajectories not only much smoother but less prone to wear out vehicle hardware, cause motion sickness, all while consuming less energy. Note how even also the deviation depicted in Figure 4.8c and Figure 4.8d is much smaller for TESC than TEB, signifying more consistent quality of results.

4.2.3 Complex obstacle scenario

The third experiment operates in a realistic scenario of a simulated small office environment depicted in Figure 4.10. The size of the grid in this case is $10,2 \times 14,85$ meters, constitutes of two distinct rooms connected by a door opening, with both rooms filled with furniture such as shelves and tables. Once again, randomly-generated goals are used to benchmark the planners while the feasibility of results is verified in a post-processing step. Statistics from the resulting metrics are presented in Figure 4.7-Figure 4.8, columns associated with this experiment are marked with **SO** suffix (Small Office).

Unlike in the circular obstacles scenario in Subsection 4.2.2, the success rate of the planners show similar tendencies, with 77% and 71% respectively for TEB and TESC. However, optimisation time of TESC slightly increased for this scenario while TEB's remained fairly low. Trajectory length and time for both planners show the same trend as for previous experiments. Similarly to previous experiments, TESC shows a smaller average velocity (Figure 4.8a) and a much smaller acceleration (Figure 4.8b)

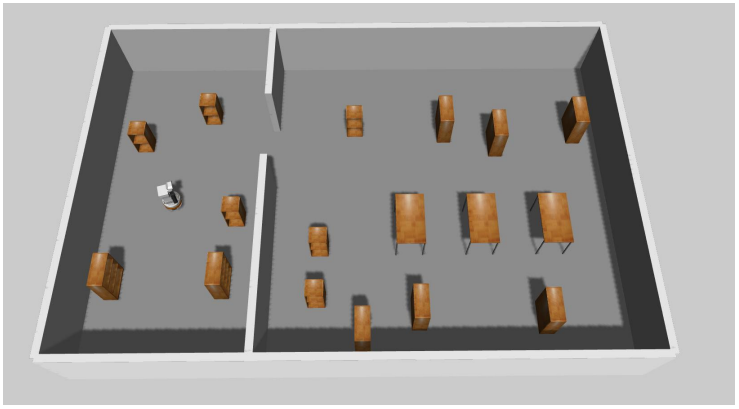


Figure 4.10. The *TIAGo* robot in the Small Office simulation environment.

than TEB. Once again, the results of TESC produce smaller and more consistent energy (Figure 4.8c) and curvature (Figure 4.8d) metrics, showing an overall improvement in the quality of output trajectories.

Implementation Details

The TESC planner has been implemented in C++ using the Ceres library [97] for solving the least-squares problem, as it is flexible and offers automatic-differentiation. The smoothness objective function relies on the recently released `manif` library [93], a Lie-theory library for state-estimation. All scenarios presented in the experiments are simulated using ROS [11], furthermore, TESC has been integrated with the navigation stack such that the output of these planners are directly applicable to a wide range of real robots using *ros_control* [79].

4.3 Conclusion

This chapter presented a novel formulation, *Timed-Elastic Smooth Curve* (TESC), for solving the motion planning problem in the mobile-base planning context. The focus of this approach and the reason for the temporary shift in domain is to explore the possibility of quality improvements using gradient-based optimisation motion planners.

This work succeeds at incorporating additional quality objectives into the optimisation process, specifically, ensuring nonvanishing *n-th* derivatives in the trajectory, allowing to constrain velocity, acceleration, jerk, etc with ease. Moreover, there is no need to add extra cost functions as the continuity of the curve at its knots is ensured by design.

Relying on a discrete set of points, this formulation allows to interpolate points on the trajectory curve. When combined with a continuous collision objective function, this property helps ensure that the entire trajectory is collision-free. The proposed approach, TESC, was benchmarked using a series of simulated mobile-base motion planning scenarios. It was shown that it prevails or matches the performance of the TEB planner in the majority of the presented metrics. TESC clearly demonstrated the expectations from this chapter's work, increasing the quality and the consistency of quality in the generated trajectories.

Despite the challenging experimental setup, both planners performed well, confirming the relevance of gradient-based optimisation planners. A possible reason for TESC not uniformly prevailing in all metrics, is due to *Euclidean Distance Grid* (EDG) operating on a rasterized representation using unsigned integers. This discretisation and unsigned-ness potentially leads to ill-shaped or non-existent gradients at obstacles, thus to optimisation issues. While not ideal, this representation was adequate for this version of TESC as it offers good computational efficiency at suboptimal theoretical performance. To improve this, a continuous distance field representation using splines could be used which would guarantee proper gradients but it was deemed to be outside of scope for this chapter. In general, while an OG-based approach integrates intuitively with optimisation methods, it will limit planning to static environments and may suffer from the issues mentioned above.

The experiments in this chapter shown that at the cost of a small increase in optimisation time, TESC is able to produce more smooth and energy-efficient trajectories of the same length as TEB but with a smaller average velocity and acceleration. The results of this chapter were published in [17].

By extending the framework to other Lie manifolds, specifically $\mathrm{SE}(3)$, arm motion planning can be tackled. Only little modification is required to do this, since Equations 4.5-4.11 are written group-agnostic, only employing the appropriate $\exp(\cdot)$ and $\log(\cdot)$ functions. Beyond supporting robot arm motion planning, such a Cartesian-space planner is also relevant for planning trajectories in other domains of robotics, *e.g.*,*Unmanned Aerial Vehicle*s (AUVs) and *Autonomous Underwater Vehicle*s (UAVs). A possible extension is to adopt *Signed Euclidean Distance Transform* for the collision avoidance objective function, allowing for the existence of discrete gradient throughout the grid and thus to recover from a collision.

Chapter 5

Timed-Elastic Bands for Manipulation Motion Planning

This chapter introduces a new method called *Timed Elastic Bands for Manipulation Motion Planning* (TEB2MP), based on the exploratory work of Chapter 4, solving the manipulation motion planning problem, while allowing to apply continuously optimised constraints to the problem during the search for a solution. Due to the nature of the presented method, it is highly extensible with new constraints and additional optimisation objectives. Specifically, this chapter is considering joint limits, path smoothness and a mixture of Cartesian and joint-space constraints at the same time.

The proposed approach is compared against state-of-the-art methods in various manipulation scenarios. Results show that this method achieves the requirement stated in the conclusion of Chapter 3, the planning is more consistent and less variant while providing a performance comparable to the state-of-the-art. This behaviour allows the proposed method to set a lower bound performance guarantee for other methods to build upon. An example planning result is shown in Figure 5.1. The initial position of the *end-effector* is on the right of the green obstacle, TEB2MP has to plan around the obstacle to reach the final position on the left. The resulting plan moves the arm to the left under the obstacle, successfully avoiding it.

The contributions of this chapter are:

- Introduction of TEB2MP, a novel trajectory optimisation method for robot arms based on *Timed Elastic Bands* (TEB).

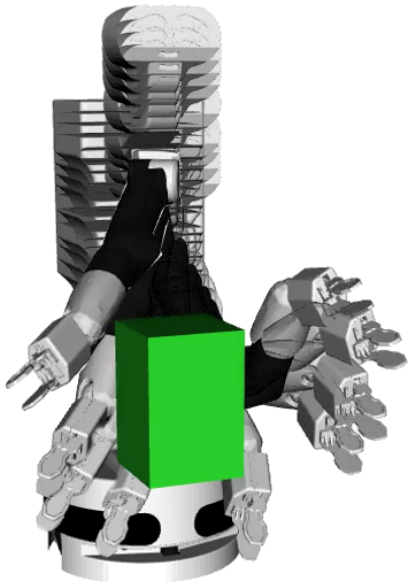


Figure 5.1. An example planning result of *TEB2MP*. The robot has to plan around the green box.

- TEB2MPs trajectory continuity, extending the formulation introduced in Section 4.1.1, a formulation of interpolation on manifolds using Lie Algebra encouraging a k-derivable Cartesian trajectory spline.
- TEB2MPs ability to handle Cartesian and joint-space constraints in a single framework.
- A collision checking method which provides smooth, continuous gradients based on overlapping volumes, ensuring that the optimisation is always able to escape colliding states.
- A rich comparison of TEB2MP and state-of-the-art methods in multiple challenging scenarios through various benchmarks.
- The planner was designed and implemented to fit into the framework proposed in Chapter 3, thus enabling it to benefit from task-informed trajectory generators.

The following paragraphs reiterate some of the state-of-the-art methods in order to place TEB2MP in their context and support design decisions.

Despite the remarkable results, the motion planning state-of-the-art still suffers from shortcomings on aspects that are hard to model and solve efficiently. For instance, smoothness is often not considered at all or doing so is ill-fitted to the nature of the planner, necessitating post-processing steps to remove jerky effects, as often done for *e.g. Rapidly-exploring Random Tree* (RRT) [41]. Although fundamental, joint limits are often naively clamped to some pre-defined ranges, creating infeasible trajectories despite an otherwise valid plan [65]. Most importantly, for general usage, handling Cartesian and joint-space constraints in a single framework is often not possible [57] or - depending on the convexity of the problem - hard to achieve robustly.

The experiments presented in [98] and Chapter 4 show that the TEB and *Timed-Elastic Smooth Curve* (TESC) approaches are capable of handling the computational complexity of nonlinear systems. The problem is formulated as a sparse graph structure solvable by means of nonlinear least-squares optimisation.

Stochastic Trajectory Optimization for Motion Planning (STOMP) [52] optimises non-differentiable constraints by stochastically adding noise to a trajectory, selecting the best from a set of such trajectories and repeating until convergence. Unfortunately, as experiments showed in Chapter 3, such methods have the tendency to be computationally inefficient as they require a well discretised trajectory to support collision checks and guarantee a smooth solution and still may fail converging on moderately hard problems. In addition, the performance of STOMP is heavily dependent on the parameters used for noise generation as experienced while preparing the experiments in Section 3.2. In many cases a set of parameters will produce good results, while the same set will perform badly in a different problem.

In Section 5.1 the proposed approach is presented in detail. Section 5.2 presents the experimental setup, along with the evaluation metrics and results. Finally, in Section 5.3 the chapter is concluded and future work is presented.

5.1 Method description

Timed Elastic Bands was originally developed by Rösmann *et al.* [99] for mobile-base navigation planning. An extension to this approach in Chapter 4 proved that consistency in results and smoothness can be achieved on top of fast planning. As detailed in Section 4.1.1, TEB formulates the problem as a sequence of $n+1$ robot poses $\in \mathrm{SE}(2)$ linked together from an initial configuration to a goal. In order to support efficient robot

arm motion planning, the optimisation domain is changed from Cartesian space to joint space.

The following present the redefinition of the TEB algorithm using a joint-space representation as the unit of optimisation, while considering constraints defined in arbitrary spaces. Consider

$$\Theta = \{\mathbf{\Phi}_0, \dots, \mathbf{\Phi}_n\} \tag{5.1}$$

the joint-space trajectory, where $\mathbf{\Phi}_i$ is the complete joint state of the robot at time i, and

$$\mathbf{q} = \{(\mathbf{\Phi_1}, \mathbf{\Delta t_1}), \dots, (\mathbf{\Phi_n}, \mathbf{\Delta t_n})\} \tag{5.2}$$

is the redefined set of state-time pairs first defined in Section 4.1.1.

Algorithm 5.1 Timed Elastic Bands for Manipulation Motion Planning

Input: $Start, Goal$
Output: $Trajectory$

$\Theta_0 \leftarrow$ GENERATEINITIALTRAJECTORY$(Start, Goal)$
$\mathbf{G} \leftarrow$ BUILDGRAPH$(\mathbf{q})$
$i \leftarrow 1$
$\mathbf{e} \leftarrow$ GETERROR$(\mathbf{G})$
while $\|\mathbf{e}\|_1 > 0$ or $i \leq NumIter$ **do**
    $\mathbf{G} \leftarrow$ NLSOPTIMISATION$(\mathbf{G})$
    $\mathbf{G} \leftarrow$ UPDATECACHEDROBOTMODEL$(\mathbf{G})$
    $\mathbf{e} \leftarrow$ GETERROR$(\mathbf{G})$
    $i \leftarrow i + 1$
$Trajectory \leftarrow$ GETTRAJECTORY$(\mathbf{G})$
return $Trajectory$

In this context, the new algorithm for TEB2MP can be written as summarised in Algorithm 5.1. Given a starting configuration $\mathbf{\Phi}_0$ and a desired *end-effector* pose $\mathbf{p}_n \in \mathrm{SE}(3)$, a goal configuration $\mathbf{\Phi}_n$ is computed by solving the inverse kinematics problem. This is done by using Trac-IK [21]. Then, an initial trajectory T_0 is created by using interpolation, generating a specific number of intermediate configurations. The experiments in this chapter employ a linear and a cubic interpolation method previously introduced in Subsection 3.1.1 but similarly to *Guided Stochastic Optimization for Motion Planning* (GSTOMP), a wide variety of trajectory generators could be used.

The following step sets up the problem as an optimisation graph $\mathbf{G}$, described in by

Kummerle *et al.* in [100]. A vertex in the presented system is a unit of data subject to optimisation. The graph is constructed using vertices representing robot joint-space configurations and time increments. These correspond to the variables the underlying numerical optimisation-solver operates on. Edges are defined connecting vertices that constrain different aspects of the trajectory (velocity, continuity, obstacle-avoidance, etc) by means of encapsulated error functions. Vertices are not required to be homogeneous, in fact, TEB2MP defines two types: a robot state vertex defined by the robot states $\mathbf{\Phi}_i$; and a time difference vertex Δt_i which defines the time difference between two consecutive robot states. The number of vertices used is the same as the number of points making up the initial trajectory Θ_0.

A caching step was introduced to support both joint-space and task-space constraints without loss of efficiency. Without this step, *end-effector* pose and internal collision models would require computation on every vertex in every optimisation step, not taking into account whether the specific vertex has changed at all. Both of the above-mentioned cached values are computed using the kinematic model and the joint-space values stored within a given vertex. The UpdateCachedRobotModel$(\mathbf{G})$ step updates a local robot model stored with each vertex using the joint-state representation denoted by $\mathbf{\Phi}_i$ and computes the corresponding Cartesian-space *end-effector* pose $\mathbf{p}_i$.

The edges of the graph describe the error functions to be minimised and the connections to vertices denote the arguments of objective functions they represent. Figure 5.2 visualizes a small example graph and illustrates a subset of edge types for readability. Vertices are drawn as circles and are connected through factors (squares) representing the various cost functions. The different objective functions are detailed in the following subsections.

5.1.1 Joint-space position, velocity and acceleration limits

Joints positions, velocities and accelerations are limited by (often physical) upper and lower boundaries translating to inequality constraints. Following the discussion in Section 4.1.1, inequality constraints are implemented by two-sided quadratic penalties defined in Equation 4.18. The definition of joint position error is inspired by the work of Tsai *et al.* in [101], specifically equation 83. It is adapted such that $P : \mathbb{R}^{dof} \rightarrow [0, 1]$ where the boundaries values 0 and 1 mark the case when the joint value is inside and outside, respectively.

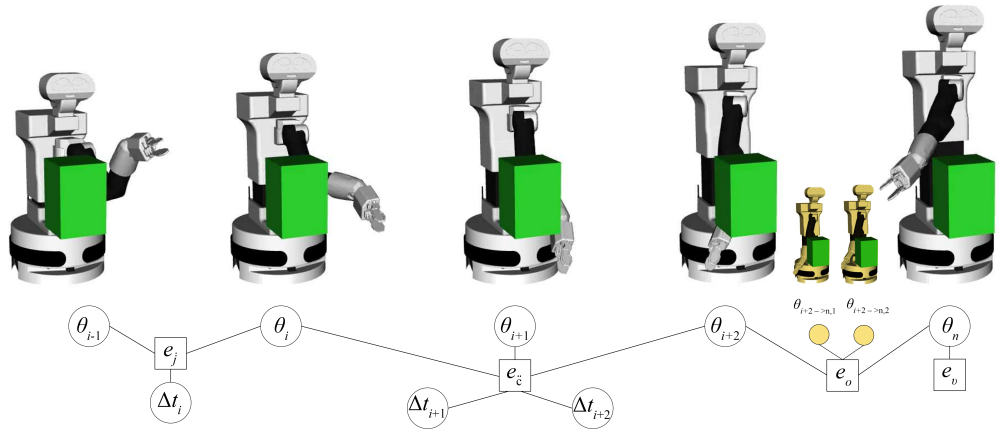


Figure 5.2. An illustration of an example TEB2MP sub-graph. Rectangles and circles depict edges and nodes respectively. Yellow mini robots are interpolated states.

$$\mathbf{e}_l = \begin{cases} 0, & \vartheta_i \in [\vartheta_i^{\mathrm{L}}, \vartheta_i^{\mathrm{U}}] \\ \|exp(\frac{(\vartheta_i - \vartheta_i^{\mathrm{L}}) * (\vartheta_i^{\mathrm{U}} - \vartheta_i)}{(\vartheta_i^{\mathrm{U}} - \vartheta_i^{\mathrm{L}})^2})\|, & \text{otherwise} \end{cases} \tag{5.3}$$

where state of a single joint is denoted by ϑ_i, which may be an angle or a position, for rotary and linear joints respectively. A complete joint state configuration is defined as $\mathbf{\Phi} = [\vartheta_0, ..., \vartheta_{\mathrm{dof}}]^T$. Joint limits for any given ϑ_i are denoted by $\vartheta_i^{\mathrm{L}}, \vartheta_i^{\mathrm{U}}$, lower and upper joint limits respectively, while $\mathbf{e}_l$ is a vector containing the individual error computed $e_{l,i}$ for each joint.

The rest of the objective equations in this chapter use the common penalty function defined in Equation 4.18. As mentioned earlier in Chapter 4, given the features of the employed optimisation framework, equality constraints are implemented by setting both lower and upper bounds of an inequality constraint to the same value.

Velocity and acceleration are obtained by using backward finite differencing through a sliding window over a section of graph $\mathbf{G}$ of, respectively, the past two and three state and time vertices.

$$\dot{\mathbf{\Phi}}_i = \frac{\mathbf{\Phi}_i - \mathbf{\Phi}_{i-1}}{\Delta t_i} \tag{5.4}$$

$$\ddot{\mathbf{\Phi}}_i = \frac{\dot{\mathbf{\Phi}}_i - \dot{\mathbf{\Phi}}_{i-1}}{\Delta t_i + \Delta t_{i-1}} \tag{5.5}$$

Accordingly, errors are defined as,

$$\mathbf{e}_{\dot{j}} = \beta(\dot{\mathbf{\Phi}}_i, \dot{\mathbf{\Phi}}^{\mathrm{L}}, \dot{\mathbf{\Phi}}^{\mathrm{U}}) \tag{5.6}$$

$$\mathbf{e}_{\ddot{j}} = \beta(\ddot{\mathbf{\Phi}}_i, \ddot{\mathbf{\Phi}}^{\mathrm{L}}, \ddot{\mathbf{\Phi}}^{\mathrm{U}}) \tag{5.7}$$

5.1.2 Cartesian-space velocity and acceleration limits

Following Equation 5.4-Equation 5.7, TEB2MP supports velocity and acceleration constraints in Cartesian-space where the $\mathrm{SE}(3)$ pose of the *end-effector* frame is computed using *Forward Kinematics* (FK).

Given such a pose $\mathbf{p}$,

$$\mathbf{p} = \begin{bmatrix} \mathbf{R} & \mathbf{r} \\ \mathbf{0} & 1 \end{bmatrix} \triangleq (\mathbf{r}, \mathbf{q}) \in \mathrm{SE}(\mathbf{3}) \tag{5.8}$$

where $\mathbf{r} \in \mathbb{R}^3$ is a position vector and $\mathbf{R}$, $\mathbf{q}$ are equivalent representations of orientation (respectively rotation matrix and unit quaternion). Velocity and acceleration terms are defined as,

$$\dot{\mathbf{p}}_i = \frac{\mathbf{p}_i - \mathbf{p}_{i-1}}{\Delta t_i} \tag{5.9}$$

$$\ddot{\mathbf{p}}_i = \frac{\dot{\mathbf{p}}_i - \dot{\mathbf{p}}_{i-1}}{\Delta t_i + \Delta t_{i-1}} \tag{5.10}$$

along with their respective errors as,

$$\mathbf{e}_{\dot{c}} = \beta(\dot{\mathbf{p}}_i, \dot{\mathbf{p}}^{\mathrm{L}}, \dot{\mathbf{p}}^{\mathrm{U}}) \tag{5.11}$$

$$\mathbf{e}_{\ddot{c}} = \beta(\ddot{\mathbf{p}}_i, \ddot{\mathbf{p}}^{\mathrm{L}}, \ddot{\mathbf{p}}^{\mathrm{U}}) \tag{5.12}$$

5.1.3 C^k Smooth Curve

Following the definitions introduced in Section 4.1.1, this subsection extends the smoothness principle from the planar world $\mathrm{SE}(2)$ of a mobile-base to the full, 3-dimensional $\mathrm{SE}(3)$ task-space of a manipulator's *end-effector*. The smoothness cost function defined in this subsection aims to push the task-space view of the optimised trajectory onto a differentiable curve on a smooth manifold.

The Lie Algebra tools used in this chapter do not form a part of the core contributions, but a comprehensive summary is included in Section A.2.

It is important to note that the scope of this chapter requires the definition of pose ($\mathbf{p}$) to be extended to elements of the group $\mathrm{SE}(3)$.

The interpolation algorithm is adapted so that, given three consecutive poses $\mathbf{p}_{i-1}$, $\mathbf{p}_i$ and $\mathbf{p}_{i+1}$ of the *end-effector* in Cartesian-space, and their associated time intervals Δt_i and Δt_{i+1}, the cost term can be described as follows: Note how the equations did not change, only the definition of its components in support for the extended domains, $\mathrm{SE}(3)$ and $\mathfrak{se}(3)$.

$$\mathbf{l}(t) = \mathbf{p}_{i-1} \oplus (t \cdot \mathbf{v}_{i-1}) \tag{5.13}$$

$$\mathbf{r}(t) = \mathbf{p}_{i+1} \oplus ((t-1) \cdot \mathbf{v}_{i+1}) \tag{5.14}$$

$$\boldsymbol{\rho}(t) = \log(\mathbf{r} \circ \mathbf{l}^{-1}) \tag{5.15}$$

$$\hat{\mathbf{p}}_i = (\phi(t) \cdot \boldsymbol{\rho}(t)) \oplus \mathbf{l} \tag{5.16}$$

with $t = \Delta t_i / (\Delta t_i + \Delta t_{i+1})$ and $\mathbf{v}_i$ is the tangent vector to the curve at point i:

$$\mathbf{v}_i = \mathbf{p}_i \ominus \mathbf{p}_{i-1} \tag{5.17}$$

Thus, the associated error vector can be defined as

$$\mathbf{e}_s = \hat{\mathbf{p}}_i \ominus \mathbf{p}_i \tag{5.18}$$

5.1.4 Collision avoidance

Collision avoidance in the context of a robot arm is made up of two parts. Self-collisions categorize all the collision event when a robot arm collides with the robot itself, arm or body parts. On some robots such collisions can be avoided by design, joint limits may not allow for self-collisions, however the majority of robot platforms cannot afford this limitation. For self-collision checks, the complete joint state of the robot is required.

The second source of collisions is with the environment of the robot. Previously in Section 4.1.1, a precomputed distance grid served as basis for the collision avoidance objective function. This is a viable approach when planning in a planar world and there exist extensions for 3-dimensional environments, however, efficient implementations can only cope with spherical or capsule-based representations of the robot body which often lack precision and require offline preprocessing. As the focus of this thesis is not solely to improve collision handling, the initial design of TEB2MP's collision avoidance

objective function uses *Axis-Aligned Bounding Box* (AABB) representations both to perform self- and environment-collision checks. This method efficiently approximates complex shapes using bounding box segments in otherwise expensive computations. The popular *Flexible Collision Library* (FCL) [75] software library is used to implement this.

TEB2MP implements collision handling as a mixed-origin cost function. It is defined as the sum of the overlapping area of the AABB with the environment and the distance of the robot with the closest obstacle (Equation 5.20). In the ideal case this provides a smooth transitioning behaviour. Similarly to TESC introduced in Chapter 4, continuity in collision checking is ensured by employing an inter-vertex sampling strategy, namely performing additional cost computations by linearly interpolating N_c configurations between two corresponding vertices, computing the cost over these states and the first vertex. The visualization of example case is shown on the right side of Figure 5.2 depicted by the small, yellow-tinted robots. The resulting collision cost function thus reads,

$$g(\mathbf{\Phi}) = \breve{V}(\mathbf{\Phi}) + \beta(\Gamma(\mathbf{\Phi}), d^{\mathrm{L}}) \tag{5.19}$$

$$\mathbf{e}_o = [g(\mathbf{\Phi}_1),\ g(\mathbf{\Phi}_{1\to 2,1}),\ ...,\ g(\mathbf{\Phi}_{1\to 2,N_c})]^T \tag{5.20}$$

where in the context of a single edge, $\mathbf{\Phi}_1, \mathbf{\Phi}_2$ are the two associated state vertices, $\mathbf{\Phi}_{1\to 2,i}$ denotes the i-th interpolated step between $\mathbf{\Phi}_1$ and $\mathbf{\Phi}_2$, $\breve{V}$ is the overlapping volume of the robot at state $\mathbf{\Phi}$, Γ is the distance to the closest point of the environment and d^{L} is a parameter denoting a minimum required distance to keep from obstacles. Although the objective function is of mixed origin, a sum of these two components was found more appropriate and numerically stable from the optimisation perspective, as combined they form a continuous gradient from colliding, through within minimum range, to collision free evaluations.

5.1.5 Cartesian Viapoint

When planning with task-specific trajectories, the initial trajectory has to be maintained as much as possible. An objective function to motivate vertices to keep the *end-effector in task space* close to the initial trajectory is defined in this section. The via point objective function operates on a point-to-point basis, allowing not only an entire trajectory to be followed but also specific points ($\mathbf{p}_v$). The cost for via point difference of the

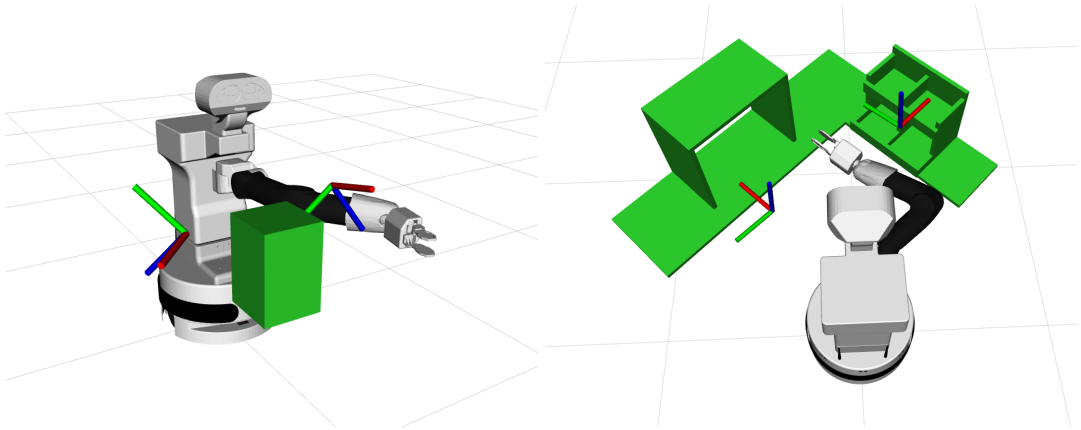
task-space pose $\mathbf{p}_v$ of the *end-effector* is defined as,

$$\mathbf{e}_v = \begin{bmatrix} \|\mathbf{r}_i, \mathbf{r}_v\| \\ d(\mathbf{q_i}, \mathbf{q_v}) \end{bmatrix} \tag{5.21}$$

where $\mathbf{p}_i = (\mathbf{r}_i, \mathbf{q_i})$ and $d(\mathbf{q_1}, \mathbf{q_2})$ denotes the unit quaternion distance defined by Ude *et al.* in [102](equation 10). In the experiments of this chapter, TEB2MP uses the via point objective function to implement goal tolerance. Similarly, if a task-informed trajectory generator was used, via point costs would be trivial to add.

5.2 Experiments

The proposed TEB2MP method was validated against a series of simulated scenarios. This section describes the experimental setup and discusses results as well.



(a) Box scenario **(b)** Complex desk scenario

Figure 5.3. Simulated scenarios used to verify TEB2MP. Initial *end-effector* conditions and desired Cartesian-space positions are marked by coordinate frames.

A rich comparison is provided with state-of-the-art planners and different setups of TEB2MP in three different simulated scenarios with the *TIAGo* robot.

The list of planners chosen to compare in these experiments is as follows:

- *Rapidly-exploring Random Tree Connect* (RRTConnect),
- STOMP with linear initialisation shown in Section 3.1.1,
- *Trajectory Optimization for Motion Planning* (TrajOpt) with linear initialisation, as above

- TEB2MP with linear initialisation, as above
- TEB2MP with cubic initialisation defined in Section 3.1.1,

The main difference between the list above and that in Section 3.2 is of *Covariant Hamiltonian Optimization for Motion Planning* (CHOMP) being replaced by TrajOpt, as Schulman *et al.* in [54] showed the latter categorically outperform CHOMP.

The experiments of this chapter use the *TIAGo* robot which was described in detail in Section 2.5. The relevant information for this chapter is that this robot platform is an 8 *Degrees of Freedom* (DoF) manipulator system: a 7 DoF arm attached to an elevating torso. As baseline, the first scenario exercised planners on solving a problem in an empty world. In the second scenario the robot has to plan a path around a free-standing box placed in front of it. It is kinematically possible for the robot to avoid the obstacle by planning around either above and below. The start and goal positions are depicted in Figure 5.3a, starting on the right side of the box, the robot has to reach the goal on the left hand side of the box. Third scenario challenges planners to perform a manipulation task in a complex desk setting first introduced as part of an industrial scenario in [65]. In this task, the robot has to reach the top of a shelf on its right-hand side and move to a bench on its left-hand side as depicted in Figure 5.3b. The implementation of TEB2MP relies on the *General Graph Optimization* (G2O) library [100] for building the problem graph and connecting it to the nonlinear least-squares solver and on the *manif* library [93] for Lie Algebra operations. All simulated scenarios were created with *Robot Operating System* (ROS) and *MoveIt*[72], and the trajectories generated by these planners are directly applicable to real robots using *ros_control* [79].

Each planner was evaluated over 100 runs in each scenario, and results were compiled into a set of statistics. Running multiple queries is required to ensure that the results are not affected by neither the runtime environment nor the probabilistic nature of some planners.

For the analysis on TEB2MP, both a linear- and a cubic-polynomial interpolation method is trialed, as defined previously in Subsection 3.1.1. The parametrisation of TEB2MP for the *TIAGo* robot is presented in Table 5.1, the same parameters were used for all experiments. Both TrajOpt and STOMP are initialised using the linear interpolation-based initialisation strategy, which generates a trajectory between the start and end configurations. This is done to see the behaviour of the proposed approach using different initialisation strategies. Furthermore, all optimisation-based planners,

Table 5.1. Parameter values used in all experiments with the *TIAGo* robot

Name	Value	Name	Value
$\mathbf{w}_l$	1	ϑ_i^{L} and ϑ_i^{U}	Defined in [103]
$\mathbf{w}_{\dot{j}}$	1	$\dot{\vartheta}_i^{\mathrm{L}}$ and $\dot{\mathbf{\Phi}}^{\mathrm{U}}$	Defined in [103]
$\mathbf{w}_{\ddot{j}}$	100	$\ddot{\vartheta}_i^{\mathrm{L}}$ and $\ddot{\mathbf{\Phi}}^{\mathrm{U}}$	0.2
$\mathbf{w}_{\dot{c}}$	100	$\dot{\mathbf{p}}^{\mathrm{L}}$ and $\dot{\mathbf{p}}^{\mathrm{U}}$	0.3
$\mathbf{w}_{\ddot{c}}$	1	$\ddot{\mathbf{p}}^{\mathrm{L}}$ and $\ddot{\mathbf{p}}^{\mathrm{U}}$	0.4
$\mathbf{w}_s$	1000	d^{L} (see Equation 5.19)	0.015
$\mathbf{w}_o$	100	N_c (see Equation 5.19)	3
$\mathbf{w}_v$	100	$NumIter$ (see Algorithm 5.1)	3

namely, TEB2MP, STOMP and TrajOpt are initialised with the same length of initial trajectories, each one having 20 points. This is done in order to provide a fair ground for comparison between these planners. To further ensure equal ground for the algorithms, experiments were realised on the same software framework implemented with *MoveIt* [72], and were executed on the same system consisting of an Intel i7-4710MQ, 2.5GHz CPU, 8GB of RAM and running on Ubuntu 16.04.

The set of metrics used for the evaluation is shown in Table 5.2. It is a revised version of Table 3.1. Most metrics from Table 4.2 are not applicable to arm planners. The first metric is the total amount of *time required* to find a valid plan, this captures the entire process for each method as if it was a query on a real robot, no planners were allowed to use pre-computed elements. The second evaluation metric is planning *success rate*. Every collision-free trajectory is considered a success, failure otherwise. The third metric captures the *smoothness* of each generated trajectory through the sum of absolute joint-space acceleration. Smoother trajectories require less acceleration or deceleration, therefore putting less stress on the mechanical parts of the robot, as well as, decreasing the energy required to execute these trajectories. It must be noted that for the smoothness metric, the lower the score the better. Lastly, the *distance from joint limits* metric highlights how far each joint is from its limits during a trajectory. Being further away from the joint limits is beneficial as plans that are close to the joint limits may be harder to execute due to hardware lock-in or precision errors. This metric was inspired by the discussions of Tsai *et al.* in [101] on the notion of manipulability.

Table 5.2. Metrics used to evaluate each experiment.

Metric	Description
Planning time in s	τ
Success rate in %	s_r
Smoothness in rad/s^2	$\ddot{\mathbf{\Phi}}$
Distance from joint limits	$\frac{\sum_{i=1}^{n}\sum_{j=1}^{m}\frac{\vartheta_{i,j}-\vartheta_{i,j}^{\min}}{\vartheta_{i,j}^{\max}-\vartheta_{i,j}^{\min}}}{n \cdot m}$

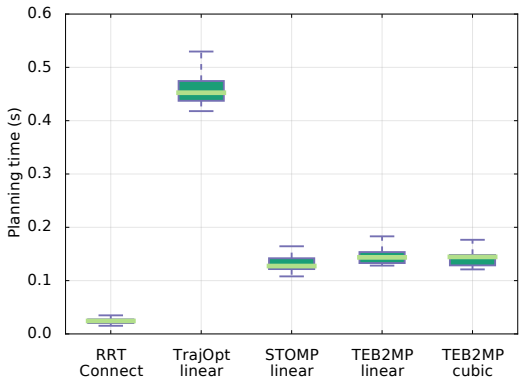
5.2.1 Empty environment

This section discusses results from the empty world scenario which acts as a baseline experiment. The start and the goal configurations for the robot are the same as for the box scenario which is depicted in Figure 5.3a, they are marked by coordinate frames.

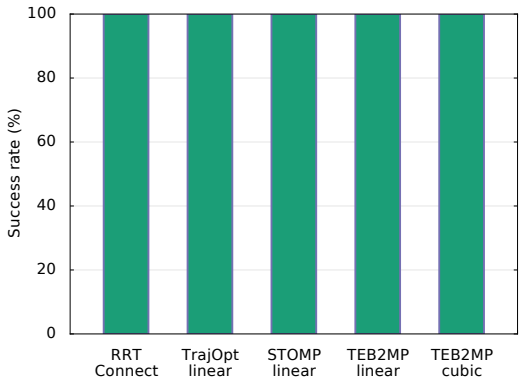
A total of 100 trials were done with the various tested planners, statistical results of these are presented in Figure 5.4. Without question, RRTConnect performs better than any other planner in the time metric. This planner was included in the experiments to serve as an example of a fast, but often low-quality planner. STOMP and TEB2MP perform approximately 6 times slower while TrajOpt comes last, being 15 times slower than RRTConnect. As expected from an empty collision scene, all planners succeeded in all attempts, producing 100% success rate. The smoothness evaluation favoured TrajOpt in this scenario. STOMP and the two variants of TEB2MP performed similarly well, while RRTConnect proved to be the least smooth. Finally, joint limit distance was best achieved by STOMP in this scenario thanks to its embedded minimum control cost bias. TEB2MP with linear initialisation is second, while cubic initialisation comes third. TrajOpt performed only slightly poorer than the cubic TEB2MP, while RRTConnect finished last again. In general, RRTConnect triumphed all other planners in planning time but lost in all other metrics, TrajOpt excelled at smoothness, STOMP at joint limit distance. The proposed TEB2MP method matched the performance of the other methods.

5.2.2 Box environment

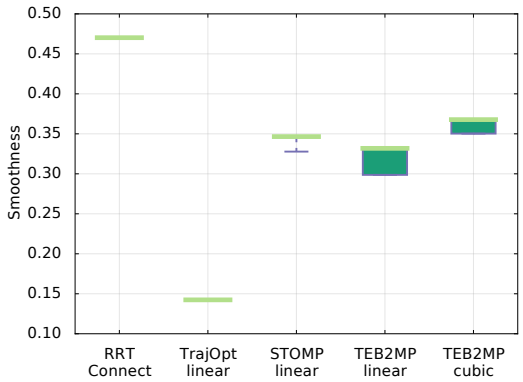
The second scenario comprises of a floating box shown in Figure 5.3a. Once again, planners were tested with 100 simulated queries in this experiment with results summarized in Figure 5.5.



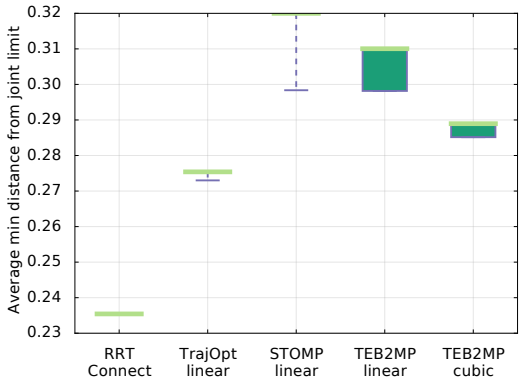
(a) Planning time, lower values are favoured



(b) Success rate, higher values are favoured



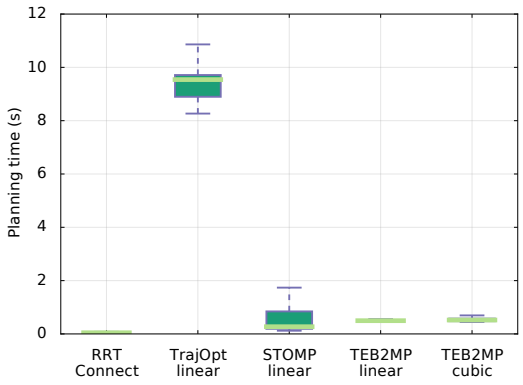
(c) Smoothness, lower values are favoured



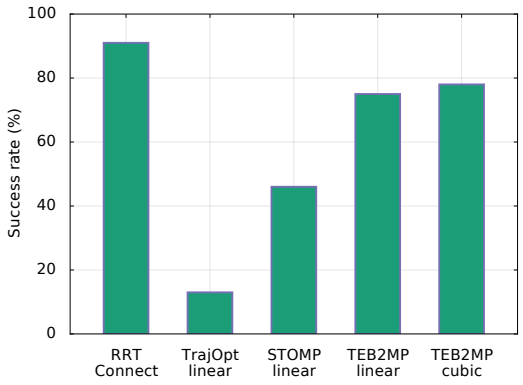
(d) Distance from joint limits, higher values are favoured

Figure 5.4. Experiment results from the empty world scenario

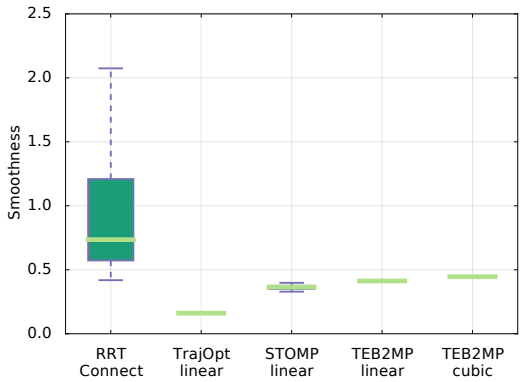
Repeating previous successes, RRTConnect performed fastest. STOMP performed better than TEB2MP in approximately 50% of the cases but did so with a very high variance, sometimes it performed much slower. Both flavours of TEB2MP required slightly more time on average but performed consistently. TrajOpt required the most amount of time to find a plan. RRTConnect was able to achieve 90% success rate with TEB2MP coming in second, with a score of almost 80%. STOMP finished with approximately 45%, while unfortunately, TrajOpt gained the lowest score. On the other hand, TrajOpt performed better on the smoothness scale than any other planner. STOMP and TEB2MP performed similarly well while RRTConnect finished with much worse results, showing a large variance in results. Finally, STOMP achieved the best score in joint limit distance with TEB2MP slightly worse but - again - with more consistency, while RRTConnect and TrajOpt came last. In general, TEB2MP produced consistent results while being among the best in class for all metrics.



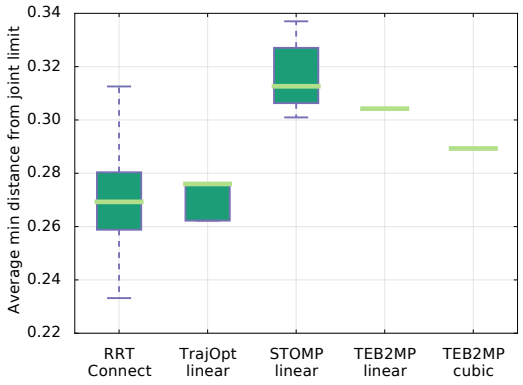
(a) Planning time, lower values are favoured



(b) Success rate, higher values are favoured



(c) Smoothness, lower values are favoured



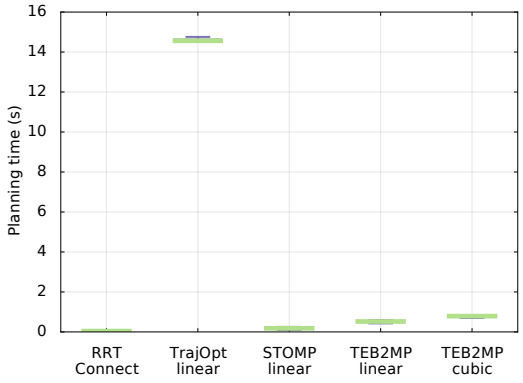
(d) Distance from joint limits, higher values are favoured

Figure 5.5. Floating box scenario experiment results

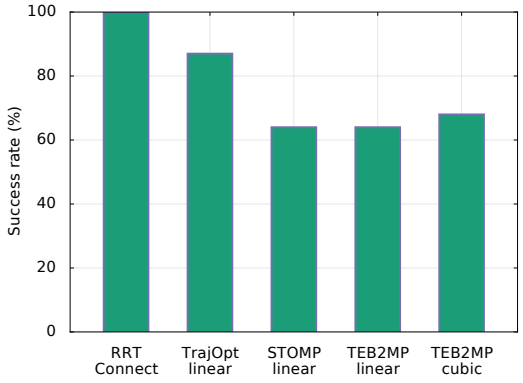
5.2.3 Complex desk environment

The most challenging scenario in the experiments depicted in Figure 5.3b is a complex desk scenario, part of an industrial setting presented in [65]. The reason for choosing this specific scenario is twofold: first, the geometry of the scenario resembles *Ambient Assisted Living* (AAL) scenes, second, it allows for a comparison discussion with the method presented in [65]. As before, statistical results were compiled in Figure 5.6.

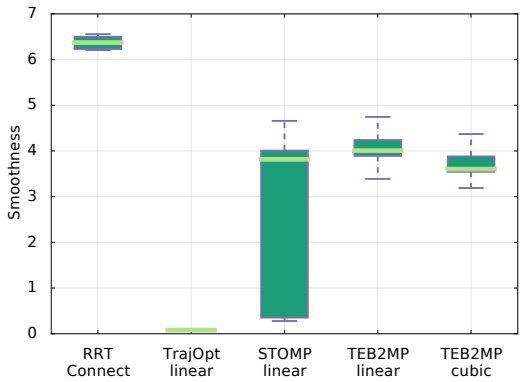
Similarly to previous experiments, the same 100 trials were ran with each planner. RRTConnect performed best in terms of planning time with STOMP second, TEB2MP following, all in a similar range. The performance of TrajOpt was greatly affected by this complex environment. RRTConnect successfully generated trajectories for all queries, achieving 100% success rate while TrajOpt performed remarkably well, with TEB2MP and STOMP following closely behind. On the smoothness criterion TrajOpt performed best, STOMP and TEB2MP variants came second, although again, STOMP



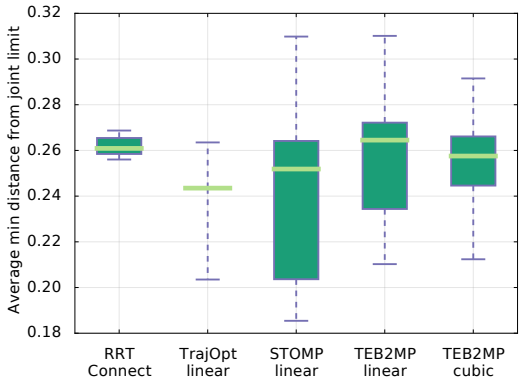
(a) Planning time, lower values are favoured



(b) Success rate, higher values are favoured



(c) Smoothness, lower values are favoured



(d) Distance from joint limits, higher values are favoured

Figure 5.6. Complex desk scenario experiment results

exhibited a very high variance on the quality of results. RRTConnect finished last in this metric. Finally, in this scenario all planners exhibited a fairly good, similar performance on joint limit distance. A possible explanation to this is that a more complex environment naturally constraints the manipulator's feasible range of motion, resulting in larger distances from limits. It is worth noting that TEB2MP consistently provided a good performance throughout this experiment as well.

Since the environment used in this experiment was first presented by Dong *et al.* in [65], and the experiments in [65] were conducted using the *PR2* robot, which is fairly similar to the *TIAGo* robot, a comparison of their solution, *Gaussian Process Motion Planner 2* (GPMP2), can be attempted. In the same environment with the *PR2* robot GPMP2 reported with an average planning time of 20ms. A caveat of these experiments is that the reported planning time did not include the time for the employed *Signed Distance Field* (SDF). For a fair comparison, the pre-processing step was measured

and added to the reported values, a GPMP2 plan takes approximately 800ms, in the same computational complexity range as TEB2MP. In terms of planning success rate, Dong *et al.* in [65] reported a single number for the *PR2* robot while two different problem sets and scenes were used. Therefore, a direct comparison cannot be made. Nevertheless, even in the case of GPMP2 having a consistently higher success rate, the optimisation process does not consider the smoothness of the trajectory or distance from joint limits.

5.3 Conclusions and future work

This chapter studies the manipulation motion planning problem in the context of optimisation-based planning for AAL environments. The proposed motion planner, TEB2MP supports multiple different types of objective functions in the optimisation. Examples include path smoothness and position, velocity and acceleration limits both in Cartesian- and joint-space. A comparison with state-of-the-art methods was made in three different scenarios of increasing difficulty. It showed a consistent behaviour with a performance comparable to the state-of-the-art for all the examined metrics. The consistency was measured through the low variance of the results and acts as a lower bound for the performance of the proposed method. In addition, TEB2MP's planning time was shown to grow linearly with the number of optimisation iterations, which is an ideal property for time-critical systems. This behaviour guarantees that even for complex problems, where more iterations are required, the total time can be kept at an upper limit. The results of this chapter were published in [18], [19].

Chapter 4 showed that a numerical gradient-based optimisation planner can provide consistency in the planning both in terms of runtime and several trajectory quality metrics. The optimisation-based arm planner presented in this chapter, TEB2MP extended this idea to arm motion planning while ensuring that task-informed trajectory generator can be introduced to the system the same way as it was shown to perform well in Chapter 3. The presented approach was shown to deliver the required characteristics for an ideal motion planning module used in an AAL context.

During the design and experiment phases it was identified that certain changes could further improve TEB2MP. The optimisation solver used currently is based on a least-squares formulation. This type of solver only allows for approximating inequalities. Equality constraints, as discussed in the context of Section 4.1.1 can be emulated by

defining two inequality constraints, one from above and one from below the desired value. Unfortunately, this may not fully satisfy the constraint as inequality constraints are always handled with a certain margin of error. This aspect of the planner could be improved by using a *Sequential Quadratic Programming* (SQP) solver using a formulation of hard constraints as defined recently by Biel *et al.* [104]. An SQP-based solver would also allow the method to generalise better if it was extended into a *Model Predictive Control* (MPC)-style formulation. Furthermore, the success rate of TEB2MP in cluttered, hard scenarios could be improved by adapting the resolution of the trajectory dynamically, such that a higher resolution (more points) is used around often colliding points, while clear, easier sections would be coarsely sampled.

The impact of introducing different initialisation strategies would be an interesting topic to study. For instance, joint- or Cartesian-space trajectories generated by other, suboptimal motion planners such as RRTConnect could be used to initialise the planner in place of the currently employed linear or cubic interpolation strategies.

The computational efficiency of motion planners is mostly dominated by the efficiency of the employed collision checking method, both at the conceptual and the implementation level. Collision distance methods, such as precomputed SDFs for example (as employed in [65]) can boost the performance at the cost of limiting the planner to static scenes and to use a task-space representation while planning. Employing an SDF approach also makes self-collision checking problematic. Locally updated collision environments have the benefit of directly querying the environment, allowing for temporal differences in the environment as well as any-time self-collision checking.

A natural extension to TEB2MP would take some ideas from the original TEB implementation, and introduce dynamic information for both obstacles and goals. The notion of time is already captured in the time vertices, this can be used to extrapolate the state of other obstacles/agents in the future. Given a highly coupled dynamic system, limiting velocity and acceleration in the manipulator ensures a friendly handover between the motion planner and a controller module. This can be achieved by, for instance, incorporating an approach to handle complex dynamics, for instance that presented by Baizid *et al.* in [105].

Chapter 6

Conclusion and Future Work

This thesis presented novel approaches for task motion planning in the context of *Ambient Assisted Living* (AAL) and small-to-medium warehousing. First, the main research questions and their proposed solutions are reiterated. This is followed by highlighting major findings, as well as some further steps for additional improvements I considered.

The original research questions were collapsed into two main questions:

- How can *Learning from Demonstration* (LfD) be used for robot motion planning?
- What type of properties can skill-informed, optimisation-based motion planning algorithms provide?

The first question was addressed by proposing the use of LfD in combination with an optimisation-based motion planner. Task demonstrations can be recorded in a robot-agnostic manner and used to generate task-specific trajectories. To acquire such demonstrations, a task-space LfD method was implemented and data was recorded by means of kinesthetic teaching. An existing stochastic optimisation-based motion planner, *Stochastic Trajectory Optimization for Motion Planning* (STOMP), was extended with the task-informed trajectory generator. The initial trajectory of the optimisation-based planner was defined by the output of the generator. The proposed motion planner showed the ability to generalise task-specific knowledge of typical *Ambient Assisted Living* (AAL) manipulation tasks such as pick and place, drawer- and cupboard opening while providing performance comparable to state-of-the-art in other scenarios. Skill-transfer between two, kinematically different robot platforms was demonstrated as well as a study on the efficiency of the proposed approach when the *Degrees of Freedom* (DoF) of a robot was changed.

Regarding the second question, there are approaches in the literature for optimisation-based motion planners that pose different qualities on the categories of computational efficiency, number and sensitivity of parameters, types of supported trajectory constraints and types of collision avoidance approaches. A motion planner that support smoothness constraints was developed using nonlinear least-squares optimisation for mobile-base navigation. This smaller problem only involves a mobile base with 2D position and heading, and it was solved using a smoothness formulation using Lie Algebra, velocity, acceleration and jerk limits, as well as *Signed Distance Field* (SDF) for collision avoidance. It was evaluated against similar state-of-the-art methods and proved superior in quality while comparable in efficiency. Based on the success of the base planner, a motion planner for robot arms was created. This extended version showed the flexibility of the least-squares optimisation in terms of supported constraints without the need for changing parameters between scenarios, demonstrating better and more consistent results. The resulting arm planner was designed with the possibility of incorporating LfD in mind, making it applicable to generalised task-specific planning over different robot platforms.

Given the results, it was shown that it is possible to incorporate task-specific knowledge into the motion planning process. This was achieved by using a combination of an optimisation-based motion planner and LfD to provide the initial trajectory for the optimiser. The original approach of stochastic optimisation was replaced by a least-squares, gradient optimisation planner in order to improve the consistency of planning results and support higher quality trajectories.

6.1 Major findings

Experimental results showed that such a planning framework can compete both in efficiency and trajectory quality with state-of-the-art planning algorithms not taking task constraints into account. The following list highlights the major findings of this work:

1. Optimisation-based planners are ideal for incorporating prior knowledge into a motion-planning system. While different optimisation methods have different benefits, it is possible to further constrain them by prior data from LfD methods without significantly sacrificing performance [16], [62].

2. Capturing and reparametrising task-specific trajectories in task-space allows an optimisation-based planner to scale well with both the number of tasks and number of robot platforms [16].

3. Gradient-based optimisation algorithms can be made to support both task- and joint-space constraints on smoothness, limits and collision avoidance while providing consistent results [17], [18]. Stochastic optimisation approaches require fine parameter tuning to avoid struggling with strict constraints.

Even though the experimental tasks were inspired by challenges posed in the AAL and logistics domains, the results strongly suggest that optimisation-based planners hold a lot of potential for future work in motion planning. Furthermore, as also pointed out by Schulman et al. in [54], initialisation plays a fundamental role in optimisation-based motion planning, which has the potential to yield further advancement to the field.

6.2 Future work

Following the findings and demonstrated performance of the proposed approaches, several candidates were identified for further extension and improvement.

1. The current implementation uses a *Dynamical Movement Primitive* (DMP)-based task-informed trajectory generator. While drop-in replacements could be implemented with any other LfD method meeting the same requirements, I think a *Probabilistic Motion Primitives* (ProMPs)-based implementation would conceptually improve on the proposed system due to its support for multiple demonstrations of a given task.

2. Currently, generated trajectories are executed directly in position-mode on the robot. Real-life, robust deployment of such planners would require a compliant controller to execute the trajectories. A DMP-based compliant controller framework such as [106] could be utilised for this, receiving the parameters of the DMP in the planner component.

3. The graph-optimisation-based method proposed in this work could be extended to support hard constraints. Alternatively, a faster, less constrained graph- or stochastic-optimisation could be combined with a slightly slower, more constrained initialiser step and be executed in an *Model Predictive Control* (MPC) scheme.

4. Regarding both optimisation-based planners, it would be beneficial to further study the effect of initialisation in general. Both LfD approaches and sampling-based planners could be utilised for this. This idea was already expressed by Schulman *et al.* in [54] as well.

5. The work presented in Chapter 4 and Chapter 5 could be combined into a joint approach where arm and mobile-base motion planning is handled in a holistic manner. The graph-based optimisation does not impose limitations on the state space, a combination of mobile-base location and body-joint state is possible to be used at once.

6.3 Summary

The work presented in this thesis showed how manipulation challenges arising in AAL can be solved by incorporating *Learning from Demonstration* into existing, optimisation-based motion planning algorithms. Resulting algorithms were compared against state-of-the-art methods in a series of everyday tasks, such as shelf-picking or opening drawers and cupboards. Special care was taken to maintain certain qualities, helping robots share a space with people. Specifically, smoothness and simultaneous joint- and task-space velocity, acceleration and jerk limit enforcement was handled to make the movement of robots more predictable , at the same time, imposing less stress on the hardware. While different optimisation methods have different benefits, it is possible to build them into systems that both generalise and scale well with the number of tasks and number of robot platforms.

It is my firm belief, that robotic manipulation can benefit greatly from general, optimisation-based motion planning algorithms leveraging demonstrations of skills. I hope that this thesis can serve as a step towards introducing more data-driven strategies into motion planning, all the while, allowing the field to break up with hand-crafted solutions and instead rely on efficient, scalable and more universal approaches.

Appendices

Appendix A

Tools from Lie Algebra

This thesis leverages tools from Lie Algebra for defining objective functions for Cartesian-space trajectory smoothness. The goal of this appendix is to provide a detailed summary of the aforementioned Lie tools, extending the concepts introduced in Section 2.2. This alternative representation defines the trajectory as a sequence of points lying on a Lie manifold $\mathrm{SE}(n)$ of rigid motions. While the mathematical tools used in this work rely on linear algebra, rotations are not part of a *vector space*. This motivates the use of Lie groups and their associated Lie Algebra, which *is* a vector space. Specifically, Chapter 4 used $\mathrm{SE}(2)$ while Chapter 5 used $\mathrm{SE}(3)$. The formulation for each spaces are presented in this appendix in a self-contained form. The reader may refer to [93] for a comprehensive summary on Lie theory and the most commonly used Lie groups in robotics.

A.1 Lie tools for $\mathrm{SE}(2)$

A.1.1 The $\mathrm{SE}(2)$ Lie group

$\mathrm{SE}(2)$ is the group of rigid motions in the plane with generic components $\left[\begin{smallmatrix}\mathbf{R} & \mathbf{r}\\ \mathbf{0} & 1\end{smallmatrix}\right] \in \mathrm{SE}(2)$ where $\mathbf{p}$ is a pose, $\mathbf{r}$ is the translation and θ angle with $\mathbf{R} = \mathbf{R}(\theta)$ being the 2D rotation component. For the sake of brevity, $\mathbf{p}$ is directly express as $\mathbf{p} = (x, y, \theta) \triangleq (\mathbf{r}, \theta)$. Further in this chapter, the operators composition $\cdot$ and inversion $^{-1}$ are defined respectively as

$$\mathbf{p}_a \cdot \mathbf{p}_b = \begin{bmatrix} \mathbf{r}_a + \mathbf{R}_a \mathbf{r}_b \\ \theta_a + \theta_b \end{bmatrix}, \qquad \mathbf{p}^{-1} = \begin{bmatrix} -\mathbf{R}^\top \mathbf{r} \\ -\theta \end{bmatrix}. \tag{A.1}$$

A.1.2 Exponential map

Associated to any point of $\mathrm{SE}(2)$ is a tangent space. The Lie Algebra of interest defined as the tangent space at the identity and noted $\mathfrak{se}(2)$[1]. The elements of this space are defined as $\begin{bmatrix} [\theta]_\times & \mathbf{u} \\ \mathbf{0} & 0 \end{bmatrix}$ with $\mathbf{u} = [u, v]^T$ and $[\theta]_\times \triangleq \begin{bmatrix} 0 & -\theta \\ \theta & 0 \end{bmatrix}$, and is isomorphic to the Cartesian space $\mathbb{R}^3$. Therefore, tangent elements can be expressed as $\mathbf{v} = (\mathbf{u}, \theta) \in \mathbb{R}^{\mathbf{3}} \simeq \mathfrak{se}(\mathbf{2})$ with the exponential map relating $\mathbb{R}^3$ and $\mathrm{SE}(2)$ defined as follows,

$$\mathbf{v} \in \mathbb{R}^{\mathbf{3}} \simeq \mathfrak{se}(\mathbf{2}) \underset{\log(\cdot)}{\overset{\exp(\cdot)}{\rightleftharpoons}} \mathbf{p} \in \mathrm{SE}(\mathbf{2}) \; . \tag{A.2}$$

Given $\mathbf{p} = (\mathbf{r}, \theta)$ and $\mathbf{v} = (\mathbf{u}, \theta)$, the exponential and logarithmic maps presented by Eade *et al.* in [107] are,

$$\exp(\mathbf{v}) \triangleq \begin{bmatrix} \mathbf{R}(\theta) & \mathbf{A} \cdot \mathbf{u} \\ \mathbf{0} & 1 \end{bmatrix} \simeq \begin{bmatrix} \mathbf{A} \cdot \mathbf{u} \\ \theta \end{bmatrix} \tag{A.3}$$

$$\log(\mathbf{p}) \triangleq \begin{bmatrix} \mathbf{A}^{-1} \cdot \mathbf{r} \\ \theta \end{bmatrix} \; . \tag{A.4}$$

where $\mathbf{R}(\theta)$ is the $2D$ rotation matrix built from angle θ and

$$\mathbf{A} = \frac{1}{\theta} \begin{bmatrix} \sin(\theta) & -(1 - \cos(\theta)) \\ 1 - \cos(\theta) & \sin(\theta) \end{bmatrix} \; . \tag{A.5}$$

A.1.3 Plus and Minus

Addition and subtraction allow us to express variations around the manifold elements $\mathbf{p}$ as vectors $\mathbf{v}$ in the tangent space. They are defined as the operators $\oplus$ and $\ominus$, such that, for $\mathbf{p}_a, \mathbf{p}_b \in \mathrm{SE}(2)$ and $\mathbf{v} = \mathbb{R}^{\mathbf{3}}$,

$$\mathbf{p}_b = \mathbf{p}_a \oplus \mathbf{v} \;\; \triangleq \mathbf{p}_a \cdot \exp(\mathbf{v}) \quad \in \mathrm{SE}(\mathbf{2}) \; , \tag{A.6}$$

$$\mathbf{v} = \mathbf{p}_b \ominus \mathbf{p}_a \triangleq \log(\mathbf{p}_a^{-1} \cdot \mathbf{p}_b) \;\; \in \mathbb{R}^3 \; . \tag{A.7}$$

These operators are respectively called right- plus and minus.

[1] It is common practice to use identity as it is the only element a group is guaranteed to have and also tangent spaces defined by elements of a group are all isomorphic.

A.1.4 Weigthed inner product

On $\mathfrak{se}(2)$ the weigthed Euclidean inner product is defined as

$$\langle \mathbf{v}, \mathbf{v} \rangle = \mathbf{v}^{\mathbf{T}} \cdot \mathbf{W} \cdot \mathbf{v} \tag{A.8}$$

with $\text{diag}(\mathbf{W}) = [1, 1, 2]$ the weight matrix relative to the space basis.

A.2 Lie tools for SE(3)

The associated tangent space $\mathfrak{se}(3)$ is made up of elements expressed in the isomorphic *Cartesian space* such that $\mathbf{v} \in \mathbb{R}^{\mathbf{6}}$. This simplification is possible since this work does not require all the tools of the Lie Algebra $\mathfrak{se}(3)$. Moreover it allows to handle regular vectors in $\mathbb{R}^3$, compatible with common linear algebra tools. Translating between the two spaces, $\mathbb{R}^6$ Lie Algebra $\mathfrak{se}(3)$ and Lie group SE(3) is achieved using the following:

$$\mathbf{v} \in \mathbb{R}^{\mathbf{6}} \underset{\simeq}{\overset{\simeq}{\rightleftharpoons}} \mathfrak{se}(\mathbf{3}) \underset{\log(\cdot)}{\overset{\exp(\cdot)}{\rightleftharpoons}} \mathbf{p} \in \text{SE}(3) \tag{A.9}$$

with $\log(\cdot) : \mathfrak{se}(3) \to \mathbb{R}^6$ and $\exp(\cdot) : \mathbb{R}^6 \to \mathfrak{se}(3)$.

The operators $\oplus$ and $\ominus$ are defined similarly to Subsection A.1.3.

$$\mathbf{p} \oplus \mathbf{v} = \mathbf{p} \circ \exp(\mathbf{v}) \tag{A.10}$$

$$\mathbf{v} \oplus \mathbf{p} = \exp(\mathbf{v}) \circ \mathbf{p} \tag{A.11}$$

$$\mathbf{p}_a \ominus \mathbf{p}_b = \log(\mathbf{p}_b^{-1} \circ \mathbf{p}_a) \tag{A.12}$$

A.3 Summary

This appendix presented the toolset required from Lie theory to define the quality constraints used in Chapter 4 and Chapter 5. A comprehensive introduction to this topic is given by Sola *et al.* in [93].

Bibliography

[1] T. Rühr, J. Sturm, D. Pangercic, M. Beetz, and D. Cremers, "A generalized framework for opening doors and drawers in kitchen environments", in *2012 IEEE International Conference on Robotics and Automation*, IEEE, 2012, pp. 3852–3858.

[2] A. Paolillo, K. Chappellet, A. Bolotnikova, and A. Kheddar, "Interlinked visual tracking and robotic manipulation of articulated objects", *IEEE Robotics and Automation Letters*, vol. 3, no. 4, pp. 2746–2753, 2018.

[3] *Care-o-bot® website*, https://www.care-o-bot-4.de, Accessed: 2019-08-30.

[4] S. Cousins, "Ros on the pr2 [ros topics]", *IEEE Robotics & Automation Magazine*, vol. 17, no. 3, pp. 23–25, 2010.

[5] B. Graf, U. Reiser, M. Hägele, K. Mauz, and P. Klein, "Robotic home assistant care-o-bot® 3-product vision and innovation platform", in *2009 IEEE Workshop on Advanced Robotics and its Social Impacts*, IEEE, 2009, pp. 139–144.

[6] M. Wise, M. Ferguson, D. King, E. Diehr, and D. Dymesich, "Fetch and freight: Standard platforms for service robot applications", in *Workshop on autonomous mobile service robots*, 2016.

[7] J. Stückler, M. Schwarz, and S. Behnke, "Mobile manipulation, tool use, and intuitive interaction for cognitive service robot cosero", *Frontiers in Robotics and AI*, vol. 3, p. 58, 2016.

[8] J. Pages, L. Marchionni, and F. Ferro, "Tiago: The modular robot that adapts to different research needs", in *International workshop on robot modularity, IROS*, 2016.

[9] T. Yamamoto, T. Nishino, H. Kajima, M. Ohta, and K. Ikeda, "Human support robot (hsr)", in *ACM SIGGRAPH 2018 Emerging Technologies*, ACM, 2018, p. 11.

[10] *Ros robots collection*, https://robots.ros.org/, Accessed: 2019-08-30, License: Creative Commons Attribution 3.0.

[11] M. Quigley, K. Conley, B. Gerkey, J. Faust, T. Foote, J. Leibs, R. Wheeler, and A. Y. Ng, "Ros: An open-source robot operating system", in *ICRA workshop on open source software*, Kobe, Japan, vol. 3, 2009, p. 5.

[12] T. Foote, *Ros community metrics report*, 2018. [Online]. Available: http://download.ros.org/downloads/metrics/metrics-report-2018-07.pdf.

[13] Y. Karayiannidis, C. Smith, F. E. V. Barrientos, P. Ögren, and D. Kragic, "An adaptive control approach for opening doors and drawers under uncertainties", *IEEE Transactions on Robotics*, vol. 32, no. 1, pp. 161–175, 2016.

[14] H. Dang and P. K. Allen, "Robot learning of everyday object manipulations via human demonstration", in *2010 IEEE/RSJ International Conference on Intelligent Robots and Systems*, IEEE, 2010, pp. 1284–1289.

[15] C. Rösmann, W. Feiten, T. Wösch, F. Hoffmann, and T. Bertram, "Efficient trajectory optimization using a sparse model", in *2013 European Conference on Mobile Robots*, Sep. 2013, pp. 138–143. DOI: 10.1109/ECMR.2013.6698833.

[16] B. Magyar, N. Tsiogkas, B. Brito, M. Patel, D. Lane, and S. Wang, "Guided stochastic optimization for motion planning", *Frontiers in Robotics and AI*, vol. 6, p. 105, 2019.

[17] J. Deray, B. Magyar, J. Solà, and J. Andrade-Cetto, "Timed-elastic smooth curve optimization for mobile-base motion planning", in *2019 IEEE/RSJ International Conference on Intelligent Robots and Systems (IROS)*, Nov. 2019, pp. 3143–3149. DOI: 10.1109/IROS40897.2019.8968240.

[18] B. Magyar, N. Tsiogkas, J. Deray, S. Pfeiffer, and D. Lane, "Timed-elastic bands for manipulation motion planning", *IEEE Robotics and Automation Letters*, vol. 4, no. 4, pp. 3513–3520, Oct. 2019, ISSN: 2377-3766. DOI: 10.1109/LRA.2019.2927956.

[19] B. Magyar, N. Tsiogkas, J. Deray, S. Pfeiffer, and D. Lane, "Timed-elastic bands for manipulation motion planning", *Intelligent Robots and Systems (IROS), 2019 IEEE/RSJ International Conference on*, 2019.

[20] R. Smits, H. Bruyninckx, and E. Aertbeliën, *Kdl: Kinematics and dynamics library (2001)*, 2013.

[21] P. Beeson and B. Ames, "Trac-ik: an open-source library for improved solving of generic inverse kinematics", in *IEEE-RAS International Conference on Humanoid Robots*, vol. 2015-Decem, 2015, pp. 928–935, ISBN: 9781479968855. DOI: 10.1109/HUMANOIDS.2015.7363472.

[22] N. Vahrenkamp, D. Muth, P. Kaiser, and T. Asfour, "Ik-map: An enhanced workspace representation to support inverse kinematics solvers", in *2015 IEEE-RAS 15th International Conference on Humanoid Robots (Humanoids)*, IEEE, 2015, pp. 785–790.

[23] R. Diankov, "Automated construction of robotic manipulation programs", PhD thesis, Robotics Institute, Carnegie Mellon University, Pittsburgh, PA, 2010.

[24] A. Billard, S. Calinon, R. Dillmann, and S. Schaal, "Survey: Robot programming by demonstration", *Handbook of robotics*, vol. 59, no. BOOK_CHAP, 2008.

[25] S. Calinon, F. Guenter, and A. Billard, "On learning, representing, and generalizing a task in a humanoid robot", *IEEE Transactions on Systems, Man, and Cybernetics, Part B (Cybernetics)*, vol. 37, no. 2, pp. 286–298, 2007.

[26] F. Zacharias, C. Borst, M. Beetz, and G. Hirzinger, "Positioning mobile manipulators to perform constrained linear trajectories", in *Intelligent Robots and Systems, 2008. IROS 2008. IEEE/RSJ International Conference on*, IEEE, 2008, pp. 2578–2584.

[27] N. Vahrenkamp, T. Asfour, and R. Dillmann, "Robot placement based on reachability inversion", in *Robotics and Automation (ICRA), 2013 IEEE International Conference on*, IEEE, 2013, pp. 1970–1975.

[28] M. Kyrarini, M. A. Haseeb, D. Ristić-Durrant, and A. Gräser, "Learning of object manipulation task actions from human demonstrations", *Mechanical Engineering*, vol. 15, no. 2, pp. 217–229, 2017. DOI: 10.22190/FUME170515010K.

[29] A. J. Ijspeert, J. Nakanishi, H. Hoffmann, P. Pastor, and S. Schaal, "Dynamical movement primitives: Learning attractor models for motor behaviors", *Neural Computation*, vol. 25, no. 2, pp. 328–373, 2013.

[30] A. Kramberger, A. Gams, B. Nemec, and A. Ude, "Generalization of orientational motion in unit quaternion space", in *IEEE-RAS International Conference on Humanoid Robots*, 2016, pp. 808–813, ISBN: 9781509047185. DOI: 10.1109/HUMANOIDS.2016.7803366.

[31] A. Ude, "Filtering in a unit quaternion space for model-based object tracking", *Robotics and Autonomous Systems*, vol. 28, no. 2, pp. 163–172, 1999, ISSN: 09218890. DOI: 10.1016/S0921-8890(99)00014-7.

[32] S. Niekum, S. Osentoski, G. Konidaris, S. Chitta, B. Marthi, and A. G. Barto, "Learning grounded finite-state representations from unstructured demonstrations", *The International Journal of Robotics Research*, vol. 34, no. 2, pp. 131–157, 2015.

[33] C. Mandery, Ö. Terlemez, M. Do, N. Vahrenkamp, and T. Asfour, "Unifying representations and large-scale whole-body motion databases for studying human motion", *IEEE Transactions on Robotics*, vol. 32, no. 4, pp. 796–809, 2016.

[34] E. Theodorou, J. Buchli, and S. Schaal, "Learning policy improvements with path integrals", in *Proceedings of the Thirteenth International Conference on Artificial Intelligence and Statistics*, 2010, pp. 828–835.

[35] F. Stulp, E. A. Theodorou, and S. Schaal, "Reinforcement learning with sequences of motion primitives for robust manipulation", *Robotics, IEEE Transactions on*, vol. 28, no. 6, pp. 1360–1370, 2012.

[36] F. Stulp and O. Sigaud, "Robot skill learning: from reinforcement learning to evolution strategies", *Paladyn, Journal of Behavioral Robotics*, vol. 4, no. 1, pp. 49–61, 2013, ISSN: 2081-4836. DOI: 10.2478/pjbr-2013-0003.

[37] A. Paraschos, C. Daniel, J. Peters, and G. Neumann, "Using probabilistic movement primitives in robotics", *Autonomous Robots*, vol. 42, no. 3, pp. 529–551, 2018.

[38] C. Paxton, G. D. Hager, L. Bascetta, *et al.*, "An incremental approach to learning generalizable robot tasks from human demonstration", in *2015 IEEE international conference on robotics and automation (ICRA)*, IEEE, 2015, pp. 5616–5621.

[39] J. Rey, K. Kronander, F. Farshidian, J. Buchli, and A. Billard, "Learning motions from demonstrations and rewards with time-invariant dynamical systems based policies", *Autonomous Robots*, vol. 42, no. 1, pp. 45–64, 2018.

[40] R. Lioutikov, G. Maeda, F. Veiga, K. Kersting, and J. Peters, "Inducing probabilistic context-free grammars for the sequencing of movement primitives", in *2018 IEEE International Conference on Robotics and Automation (ICRA)*, IEEE, 2018, pp. 1–8.

[41] J. J. Kuffner and S. M. LaValle, "RRT-connect: An efficient approach to single-query path planning", in *Robotics and Automation, 2000. Proceedings. ICRA'00. IEEE International Conference on*, IEEE, vol. 2, 2000, pp. 995–1001.

[42] S. M. LaValle and J. J. Kuffner Jr, "Randomized kinodynamic planning", *The international journal of robotics research*, vol. 20, no. 5, pp. 378–400, 2001.

[43] L. Kavraki, P. Svestka, J.-C. Latombe, and M. Overmars, "Probabilistic roadmaps for path planning in high-dimensional configuration spaces", *IEEE Transactions on Robotics and Automation*, vol. 12, no. 4, pp. 566–580, 1996.

[44] S. Karaman and E. Frazzoli, "Sampling-based algorithms for optimal motion planning", *The international journal of robotics research*, vol. 30, no. 7, pp. 846–894, 2011.

[45] D. J. Webb and J. Van Den Berg, "Kinodynamic rrt*: Asymptotically optimal motion planning for robots with linear dynamics", in *2013 IEEE International Conference on Robotics and Automation*, IEEE, 2013, pp. 5054–5061.

[46] O. Khatib, "Real-time obstacle avoidance for manipulators and mobile robots", in *IEEE International Conference on Robotics and Automation*, vol. 2, Mar. 1985, pp. 500–505, ISBN: 0818606150. DOI: 10.1109/ROBOT.1985.1087247.

[47] N. Ratliff, M. Zucker, J. A. Bagnell, and S. Srinivasa, "Chomp: Gradient optimization techniques for efficient motion planning", in *Robotics and Automation, 2009. ICRA'09. IEEE International Conference on*, IEEE, 2009, pp. 489–494.

[48] M. Zucker, N. Ratliff, A. D. Dragan, M. Pivtoraiko, M. Klingensmith, C. M. Dellin, J. A. Bagnell, and S. S. Srinivasa, "Chomp: Covariant hamiltonian optimization for motion planning", *The International Journal of Robotics Research*, vol. 32, no. 9-10, pp. 1164–1193, 2013.

[49] A. Byravan, B. Boots, S. S. Srinivasa, and D. Fox, "Space-time functional gradient optimization for motion planning", in *Robotics and Automation (ICRA), 2014 IEEE International Conference on*, IEEE, 2014, pp. 6499–6506.

[50] K. He, E. Martin, and M. Zucker, "Multigrid CHOMP with local smoothing", in *Humanoid Robots (Humanoids), 2013 13th IEEE-RAS International Conference on*, IEEE, 2013, pp. 315–322.

[51] T. Osa, A. M. G. Esfahani, R. Stolkin, R. Lioutikov, J. Peters, and G. Neumann, "Guiding trajectory optimization by demonstrated distributions", *IEEE Robotics and Automation Letters*, vol. 2, no. 2, pp. 819–826, 2017.

[52] M. Kalakrishnan, S. Chitta, E. Theodorou, P. Pastor, and S. Schaal, "Stomp: stochastic trajectory optimization for motion planning", *Proceedings - IEEE International Conference on Robotics and Automation*, pp. 4569–4574, 2011, ISSN: 10504729. DOI: 10.1109/ICRA.2011.5980280.

[53] J. Schulman, J. Ho, A. X. Lee, I. Awwal, H. Bradlow, and P. Abbeel, "Finding locally optimal, collision-free trajectories with sequential convex optimization.", in *Robotics: Science and Systems*, Citeseer, vol. 9, 2013, pp. 1–10.

[54] J. Schulman, Y. Duan, J. Ho, A. Lee, I. Awwal, H. Bradlow, J. Pan, S. Patil, K. Goldberg, and P. Abbeel, "Motion planning with sequential convex optimization and convex collision checking", *The International Journal of Robotics Research*, vol. 33, no. 9, pp. 1251–1270, 2014.

[55] D. Coleman, I. A. Şucan, M. Moll, K. Okada, and N. Correll, "Experience-based planning with sparse roadmap spanners", in *Robotics and Automation (ICRA), 2015 IEEE International Conference on*, IEEE, 2015, pp. 900–905.

[56] M. Phillips, V. Hwang, S. Chitta, and M. Likhachev, "Learning to plan for constrained manipulation from demonstrations", *Autonomous Robots*, vol. 40, no. 1, pp. 109–124, 2016.

[57] Z. Marinho, B. Boots, A. Dragan, A. Byravan, G. J. Gordon, and S. Srinivasa, "Functional gradient motion planning in reproducing kernel hilbert spaces", *Robotics: Science and Systems XII*, Jun. 2016. DOI: 10.15607/RSS.2016.XII.046.

[58] B. J. Cohen, S. Chitta, and M. Likhachev, "Search-based planning for manipulation with motion primitives", in *Robotics and Automation (ICRA), 2010 IEEE International Conference on*, IEEE, 2010, pp. 2902–2908.

[59] C. B. Browne, E. Powley, D. Whitehouse, S. M. Lucas, P. I. Cowling, P. Rohlfshagen, S. Tavener, D. Perez, S. Samothrakis, and S. Colton, "A survey of monte carlo tree search methods", *IEEE Transactions on Computational Intelligence and AI in games*, vol. 4, no. 1, pp. 1–43, 2012.

[60] D. Koert, G. Maeda, R. Lioutikov, G. Neumann, and J. Peters, "Demonstration based trajectory optimization for generalizable robot motions", in *Humanoid Robots (Humanoids), 2016 IEEE-RAS 16th International Conference on*, IEEE, 2016, pp. 515–522.

[61] S. Stark, J. Peters, and E. Rueckert, "Experience reuse with probabilistic movement primitives", *Intelligent Robots and Systems (IROS), 2019 IEEE/RSJ International Conference on*, 2019.

[62] M. A. Rana, M. Mukadam, S. R. Ahmadzadeh, S. Chernova, and B. Boots, "Towards robust skill generalization: Unifying learning from demonstration and motion planning", in *Conference on Robot Learning*, 2017, pp. 109–118.

[63] C. Rösmann, F. Hoffmann, and T. Bertram, "Kinodynamic trajectory optimization and control for car-like robots", in *2017 IEEE/RSJ International Conference on Intelligent Robots and Systems (IROS)*, Sep. 2017, pp. 5681–5686. DOI: 10.1109/IROS.2017.8206458.

[64] S. Quinlan and O. Khatib, "Elastic bands: Connecting path planning and control", in *IEEE International Conference on Robotics and Automation*, May 1993, 802–807 vol.2. DOI: 10.1109/ROBOT.1993.291936.

[65] J. Dong, M. Mukadam, F. Dellaert, and B. Boots, "Motion planning as probabilistic inference using gaussian processes and factor graphs.", in *Robotics: Science and Systems*, vol. 12, 2016.

[66] M. Bhardwaj, B. Boots, and M. Mukadam, "Differentiable gaussian process motion planning", *ArXiv preprint arXiv:1907.09591*, 2019.

[67] G. I. Boutselis, Y. Pan, and E. A. Theodorou, "Numerical trajectory optimization for stochastic mechanical systems", *SIAM Journal on Scientific Computing*, vol. 41, no. 4, A2065–A2087, 2019.

[68] H. Mao and J. Xiao, "Real-time conflict resolution of task-constrained manipulator motion in unforeseen dynamic environments", *IEEE Transactions on Robotics*, 2019.

[69] B. Calli, A. Walsman, A. Singh, S. Srinivasa, P. Abbeel, and A. M. Dollar, "Benchmarking in manipulation research: Using the yale-cmu-berkeley object and model set", *IEEE Robotics & Automation Magazine*, vol. 22, no. 3, pp. 36–52, 2015.

[70] P. R. Wurman and J. M. Romano, "The amazon picking challenge 2015 [competitions]", *IEEE Robotics Automation Magazine*, vol. 22, no. 3, pp. 10–12, Sep. 2015, ISSN: 1070-9932. DOI: 10.1109/MRA.2015.2452071.

[71] N. Correll, K. E. Bekris, D. Berenson, O. Brock, A. Causo, K. Hauser, K. Okada, A. Rodriguez, J. M. Romano, and P. R. Wurman, "Lessons from the amazon picking challenge", 2016.

[72] I. A. Sucan and S. Chitta, "Moveit!", *Online at http://moveit. ros. org*, 2013.

[73] R. Diankov and J. Kuffner, "Openrave: A planning architecture for autonomous robotics", *Robotics Institute, Pittsburgh, PA, Tech. Rep. CMU-RI-TR-08-34*, vol. 79, 2008.

[74] H. Sakoe and S. Chiba, "Dynamic programming algorithm optimization for spoken word recognition", *IEEE Trans. Acoustics, Speech, and Signal Processing*, vol. 26, no. 1, pp. 43–49, 1978, ISSN: 0096-3518. DOI: 10.1109/TASSP.1978.1163055.

[75] J. Pan, S. Chitta, and D. Manocha, "Fcl: A general purpose library for collision and proximity queries", in *Robotics and Automation (ICRA), 2012 IEEE International Conference on*, IEEE, 2012, pp. 3859–3866.

[76] A. Jain, H. Nguyen, M. Rath, J. Okerman, and C. C. Kemp, "The complex structure of simple devices: A survey of trajectories and forces that open doors and drawers", in *Biomedical Robotics and Biomechatronics (BioRob), 2010 3rd IEEE RAS and EMBS International Conference on*, IEEE, 2010, pp. 184–190.

[77] I. A. Sucan, M. Moll, and L. E. Kavraki, "The open motion planning library", *IEEE Robotics & Automation Magazine*, vol. 19, no. 4, pp. 72–82, 2012.

[78] B. Cohen, I. A. Sucan, and S. Chitta, "A generic infrastructure for benchmarking motion planners", in *Intelligent Robots and Systems (IROS), 2012 IEEE/RSJ International Conference on*, IEEE, 2012, pp. 589–595.

[79] S. Chitta, E. Marder-Eppstein, W. Meeussen, V. Pradeep, A. R. Tsouroukdissian, J. Bohren, D. Coleman, B. Magyar, G. Raiola, M. Lüdtke, *et al.*, "Ros_control: A generic and simple control framework for ROS", *The Journal of Open Source Software*, vol. 2, p. 456, 2017.

[80] Y. Meirovitch, D. Bennequin, and T. Flash, "Geometrical invariance and smoothness maximization for task-space movement generation", *IEEE Transactions on Robotics*, vol. 32, no. 4, pp. 837–853, 2016, ISSN: 15523098. DOI: 10.1109/TRO.2016.2581208.

[81] M. Stilman, "Global manipulation planning in robot joint space with task constraints", *IEEE Transactions on Robotics*, vol. 26, no. 3, pp. 576–584, 2010.

[82] M. Elbanhawi, M. Simic, and R. Jazar, "In the passenger seat: Investigating ride comfort measures in autonomous cars", *IEEE Intelligent Transportation Systems Magazine*, vol. 7, no. 3, pp. 4–17, 2015.

[83] M. Zefran, V. Kumar, and C. B. Croke, "On the generation of smooth three-dimensional rigid body motions", *IEEE Transactions on Robotics and Automation*, vol. 14, no. 4, pp. 576–589, Aug. 1998, ISSN: 1042-296X. DOI: 10.1109/70.704225.

[84] D. Mellinger and V. Kumar, "Minimum snap trajectory generation and control for quadrotors", in *IEEE International Conference on Robotics and Automation*, May 2011, pp. 2520–2525, ISBN: 9781612843865. DOI: 10.1109/ICRA.2011.5980409.

[85] H. Oleynikova, M. Burri, Z. Taylor, J. Nieto, R. Siegwart, and E. Galceran, "Continuous-time trajectory optimization for online uav replanning", *IEEE International Conference on Intelligent Robots and Systems*, J. A. Carrasco-Ochoa, J. F. Martínez-Trinidad, J. H. Sossa-Azuela, J. A. Olvera López, and F. Famili, Eds., pp. 5332–5339, Sep. 2016, ISSN: 21530866. DOI: 10.1109/IROS.2016.7759784. arXiv: 1812.01537.

[86] S. Liu, M. Watterson, K. Mohta, K. Sun, S. Bhattacharya, C. J. Taylor, and V. Kumar, "Planning dynamically feasible trajectories for quadrotors using safe flight corridors in 3-D complex environments", *IEEE Robotics and Automation Letters*, vol. 2, no. 3, pp. 1688–1695, Jul. 2017, ISSN: 2377-3766. DOI: 10.1109/LRA.2017.2663526.

[87] S. Lovegrove, A. Patron-Perez, and G. Sibley, "Spline fusion: a continuous-time representation for visual-inertial fusion with application to rolling shutter cameras", in *British Machine Vision Conference*, BMVA Press, 2014, pp. 93.1–93.11, ISBN: 1-901725-49-9. DOI: 10.5244/c.27.93.

[88] T. Mercy, R. V. Parys, and G. Pipeleers, "Spline-based motion planning for autonomous guided vehicles in a dynamic environment", *IEEE Transactions on Control Systems Technology*, pp. 1–8, 2017, ISSN: 1063-6536. DOI: 10.1109/TCST.2017.2739706.

[89] S. Jalel, P. Marthon, and A. Hamouda, "NURBS-based multi-objective path planning", in *Mexican Conference on Pattern Recognition. Lecture Notes in Computer Science*, vol. 9116, 2015, pp. 190–199, ISBN: 9783319192635. DOI: 10.1007/978-3-319-19264-2_19.

[90] S. Jalel, P. Marthon, and A. Hamouda, "A new path generation algorithm based on accurate NURBS curves", *International Journal of Advanced Robotic Systems*, vol. 13, no. 2, p. 75, 2016. DOI: 10.5772/63072.

[91] F. C. Park and B. Ravani, "Bézier curves on riemannian manifolds and lie groups with kinematics applications", *Journal of Mechanical Design*, vol. 117, no. 1, p. 36, 2008, ISSN: 10500472. DOI: 10.1115/1.2826114.

[92] T. Popiel and L. Noakes, "Bézier curves and C^2 interpolation in Riemannian manifolds", *Journal of Approximation Theory*, vol. 148, no. 2, pp. 111–127, 2007, ISSN: 0021-9045.

[93] J. Solà, J. Deray, and D. Atchuthan, "A micro lie theory for state estimation in robotics", Institut de Robòtica i Informàtica Industrial, Barcelona, Tech. Rep. IRI-TR-18-01, 2018. arXiv: 1812.01537.

[94] J. Jakubiak, F. S. Leite, and R. C. Rodrigues, "A two-step algorithm of smooth spline generation on riemannian manifolds", *Journal of Computational and Applied Mathematics*, vol. 194, no. 2, pp. 177–191, 2006, ISSN: 0377-0427. DOI: https://doi.org/10.1016/j.cam.2005.07.003.

[95] C. Richter, A. Bry, and N. Roy, "Polynomial trajectory planning for aggressive quadrotor flight in dense indoor environments", in *16th International Symposium on Robotics Research*, ser. Springer Tracts in Advanced Robotics, vol. 114, Springer, 2016, pp. 649–666.

[96] P. F. Felzenszwalb and D. P. Huttenlocher, "Distance transforms of sampled functions", *Theory of Computing*, vol. 8, no. 1, pp. 415–428, 2012.

[97] S. Agarwal, K. Mierle, *et al.*, *Ceres solver*, http://ceres-solver.org.

[98] C. Rösmann, F. Hoffmann, and T. Bertram, "Timed-elastic-bands for time-optimal point-to-point nonlinear model predictive control", in *Control Conference (ECC), 2015 European*, IEEE, 2015, pp. 3352–3357.

[99] C. Rösmann, W. Feiten, T. Wösch, F. Hoffmann, and T. Bertram, "Trajectory modification considering dynamic constraints of autonomous robots", in *Robotics; Proceedings of ROBOTIK 2012; 7th German Conference on*, VDE, 2012, pp. 1–6.

[100] R. Kümmerle, G. Grisetti, H. Strasdat, K. Konolige, and W. Burgard, "G 2 o: A general framework for graph optimization", in *Robotics and Automation (ICRA), 2011 IEEE International Conference on*, IEEE, 2011, pp. 3607–3613.

[101] M.-J. Tsai, "Workspace geometric characterization and manipulability of industrial robots", PhD thesis, The Ohio State University, Department of Mechanical Engineering, 1986.

[102] A. Ude, "Filtering in a unit quaternion space for model-based object tracking", *Robotics and Autonomous Systems*, vol. 28, no. 2-3, pp. 163–172, 1999.

[103] *Tiago robot description*, https://github.com/pal-robotics/tiago_robot, Accessed: 2020-03-03.

[104] M. Biel and M. Norrlöf, "Efficient trajectory reshaping in a dynamic environment", in *Advanced Motion Control (AMC), 2018 IEEE 15th International Workshop on*, IEEE, 2018, pp. 54–59.

[105] K. Baizid, G. Giglio, F. Pierri, M. A. Trujillo, G. Antonelli, F. Caccavale, A. Viguria, S. Chiaverini, and A. Ollero, "Experiments on behavioral coordinated control of an unmanned aerial vehicle manipulator system", in *2015 IEEE International Conference on Robotics and Automation (ICRA)*, IEEE, 2015, pp. 4680–4685.

[106] K. Vlachos and Z. Doulgeri, "A control scheme with a novel dmp-robot coupling achieving compliance and tracking accuracy under unknown task dynamics and model uncertainties", *IEEE Robotics and Automation Letters*, vol. 5, no. 2, pp. 2309–2315, 2020.

[107] E. Eade, "Lie groups for 2d and 3d transformations", Tech. Rep., 2013.